ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ

•

ИНФОРМАТИКА ДЛЯ ВСЕХ



ОБ АВТОРАХ И ПУБЛИКАЦИЯХ

Макаров Игорь Михайлович, член-корреспондент АН СССР, заместитель министра высшего и среднего специального образования СССР.

Кочетков Геннадий Борисович, кандидат экономических наук, заведующий сектором Института США и Канады АН СССР.

Брябрин Виктор Михайлович, кандидат технических наук, заведующий лабораторией программного обеспечения персональных компьютеров Вычислительного центра АН СССР, доцент кафедры алгоритмических языков факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова.

Сенин Григорий Васильевич, кандидат физико-математических наук, старший научный сотрудник лаборатории программного обеспечения персональных компьютеров Вычислительного центра АН СССР.

Мазурик Владимир Петрович, кандидат технических наук, заведующий лабораторией прикладных систем для персональных компьютеров Вычислительного центра АН СССР.

Все статьи написаны специально для сборника.

АКАДЕМИЯ НАУК СССР
СЕРИЯ "КИБЕРНЕТИКА –
НЕОГРАНИЧЕННЫЕ ВОЗМОЖНОСТИ
И ВОЗМОЖНЫЕ ОГРАНИЧЕНИЯ"

ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ



ИНФОРМАТИКА ДЛЯ ВСЕХ



МОСКВА
"НАУКА"
1987

ББК 32.973.2
П26
УДК 519.6

Рецензенты:

академик Г.С. ПОСПЕЛОВ,
В.И. ГРОМЫКО

П26

Персональные компьютеры. Информатика для всех. – М.: Наука, 1987. – 149 с., ил. – (Серия "Кибернетика – неограниченные возможности и возможные ограничения").

Сборник представляет собой популярное изложение принципов работы с одним из основных инструментов информатики – персональным компьютером. Описываются устройство персонального компьютера, основные приемы программирования на языках Бейсик и Лого, методы решения прикладных задач. Рассматриваются особенности применения персональных компьютеров в учреждениях, на производстве и дома.

Для широкого круга читателей.

П $\frac{1504000000-571}{054(02)-87}$ 52–87 НП ББК 32.973.2

Серия "Кибернетика – неограниченные возможности и возможные ограничения" удостоена диплома I степени на Всесоюзном конкурсе общества "Знание" в 1985 г.

ПРЕДИСЛОВИЕ

В Программе партии (Новая редакция), утвержденной XXVII съездом КПСС, глубоко и всесторонне обоснована задача ускорения научно-технического прогресса на современном этапе коммунистического строительства в нашей стране. Решение этой задачи предполагает "повсеместное внедрение новейших достижений науки и техники в производство, управление, сферу обслуживания и быта"[1]. Подлинным катализатором научно-технического прогресса является современная вычислительная техника. Намечено "обеспечить рост объема производства вычислительной техники в 2–2,3 раза, повысить ее надежность. Высокими темпами наращивать масштабы применения современных высокопроизводительных электронно-вычислительных машин всех классов"[2]. При этом важная роль в компьютеризации всех сфер жизни общества принадлежит микрокомпьютерам, и в частности, персональным компьютерам. В ближайшие годы предстоит "организовать массовый выпуск персональных компьютеров"[3].

Повышенное внимание к микрокомпьютерам и такой их разновидности, как персональные компьютеры, вполне закономерно. Микрокомпьютеры являются, пожалуй, самым блестящим достижением последних пятнадцати лет развития микроэлектроники. Сравнительно краткая история этих машин восходит к появлению в начале семидесятых годов микропроцессоров. Среди различных вычислительных систем, созданных на базе микропроцессоров, микрокомпьютеры сразу же выделились своими особенностями: обладая подчас не меньшей производительностью, чем громоздкие ЭВМ первых поколений, микрокомпьютеры несравненно более предпочтительны по таким показателям, как габариты, стоимость, энергопотребление, надежность. Если к этому еще добавить гибкость и приспособляемость микрокомпьютеров к самым различным задачам науки, техники и производства, то можно представить

---

[1] Программа Коммунистической партии Советского Союза: (Новая редакция). М.: Политиздат, 1986. С. 27.

[2] Основные направления экономического и социального развития СССР на 1986–1990 гг. и на период до 2000 г. М.: Политиздат, 1986. С. 23.

[3] Там же. С. 23.

себе не только широчайшую сферу применения микропроцессорной техники, но и обусловливаемый ею более высокий уровень использования современной вычислительной техники в целом.

В отличие от громоздких ЭВМ первых поколений микропроцессоры и микроЭВМ могут быть легко встроены в любое орудие производства, в любой объект, требующий оперативного управления. Речь идет не только об усовершенствовании и оптимизации работы уже известных изделий: станков, автомобилей, телевизоров и т.п., но и о создании принципиально новых аппаратов, приборов, инструментов.

Персональный компьютер (ПК), которому посвящена эта книга, стал одним из наиболее перспективных новшеств современной научно-технической революции. Использование ПК делает реальной перспективу качественно нового уровня переработки информации во всех сферах общественной жизни. Уже сейчас очевидно, что при планомерном и продуманном внедрении ПК могут внести существенный вклад в повышение производительности труда в науке и технике, производстве и управлении, проектировании и образовании. Разумеется, использование ПК в каждой из этих сфер имеет свои особенности. Богатое многообразие эффективных применений ПК будет отражено в последующих сборниках серии "Кибернетика – неограниченные возможности и возможные ограничения". Здесь хотелось бы подчеркнуть роль ПК в школьной и вузовской информатике, т.е. именно там, где закладывается прочная и широкая база компьютеризации общества.

По существу, сегодня каждый педагог должен думать о превращении ПК в своего действенного помощника. Ведь при адекватном подключении к учебному процессу ПК становится мощным усилителем интеллектуальных возможностей обучаемого, способствует повышению качества знаний, помогает более умело их применять. Поэтому при решении задач каждой ступени образования необходимо предусмотреть усвоение и развитие навыков использования компьютера. И школьник и студент должны видеть в ПК привычный инструмент оптимизации и интенсификации информационных процессов: приобретения и закрепления знаний, постановки и решения задач, проектирования и т.д.

Подготовку выпускника вуза нужно связывать с использованием компьютеров в его будущей профессиональной деятельности. Для этого ПК должен быть освоен молодым специалистом и как самостоятельный инструмент для практической работы, и как "окно" в более широкий компьютерный мир вычислительных сетей, объединяющих ЭВМ разных классов. Сложность многих прикладных задач потребует от молодого специалиста умения опираться на ресурсы мощных вычислительных систем.

При всей многоплановости проблем компьютеризации в образовании необходимо учесть, что решены они должны быть в кратчайшие сроки.

В настоящее время ощущается острая потребность в научно-популярной литературе, посвященной персональным компьютерам. Те статьи, которые собраны в этой книге, раскрывают некоторые важные аспекты этого вида вычислительной техники.

Сборник начинается с попытки историко-научного освещения развития ПК, оценки их роли как одного из факторов научно-технического прогресса. Оценка автором вклада ПК в современную экономику и культуру несомненно стимулирует интерес к дальнейшему обсуждению этих вопросов.

Вторая статья посвящена техническим вопросам: устройству ПК и принципам его работы. Отсюда читатель почерпнет самые первые сведения, начиная хотя бы с того, как выглядит ПК. Здесь собраны также различные технические данные, касающиеся персональных компьютеров, их составных частей (дисплеев, дисков и дисководов, адаптеров и т.д.) и сопутствующих устройств – "периферии" (принтеров, графопостроителей и др.). Затронуты некоторые вопросы программного обеспечения – неотъемлемого атрибута вычислительных машин, – в частности говорится об операционных системах и языках программирования. По существу, эта статья сборника является популярным введением в область персональной информатики, подготавливающим читателя к более глубокому изучению вопросов функционирования и эксплуатации микропроцессорной техники.

Третья статья посвящена самому элементарному введению в программирование. Что такое программа? Какие аналогии этого понятия мы можем обнаружить вокруг себя в нашей повседневной деятельности? Как программировать, т.е. составлять программу для ЭВМ? На эти вопросы делается попытка ответить в самом простом виде, интуитивно ясным образом. Программирование до поры до времени считалось уделом избранных, как когда-то вождение автомобиля. Наступило время, когда "водить машину", теперь уже электронно-вычислительную, должны многие – программирование становится более доступным, с ним надо активно знакомиться. Автор статьи постоянно имел в виду эту основную цель. Язык Бейсик взят за исходную точку для введения в предмет с учетом того, что это наиболее популярный сейчас язык программирования для персональных компьютеров. Наряду с этим в статье содержится краткое описание языка Лого, который начинает широко использоваться в обучении. Знакомство с методикой программирования на Бейсике и Лого может быть полезным для самых широких кругов читателей, и особенно для школьников. Прочное усвоение элементарных навыков программирования в

значительной степени предопределяет дальнейшее продвижение в этой области. В программировании есть своя высшая математика, но с "арифметикой программирования" наши школьники должны быть знакомы наравне с обычной арифметикой.

Хотя программирование на персональном компьютере весьма занимательная вещь, большая часть людей не станет программировать из любви к искусству, а захочет решать задачи из своей профессиональной сферы. Решению прикладных задач на ПК посвящена четвертая статья сборника. Здесь описаны средства (тоже программы), которые оптимизируют деятельность в наиболее информационно емких приложениях: обработка текстовой информации и подготовка документов, ведение взаимозависимых данных в табличном формате, хранение больших объемов однотипной информации и манипулирование ими. Для пользования этими программными инструментами нужно быть программистом лишь чуть-чуть, чтобы осознавать, "что происходит". Нечто аналогичное наблюдается при управлении автомашиной – от водителя не требуется умения разбираться в тонкостях работы мотора. Читатель убедится, что даже непрограммирующий пользователь может успешно решать достаточно сложные и интересные задачи.

В таких областях, научная популяризация которых только начинается, нельзя требовать от первых же книг исчерпывающей полноты и законченности. В следующих сборниках понадобится более детальное изложение задач программирования. В частности, потребуется рассмотреть вопросы представления структур данных, некоторые базовые алгоритмы, дать сравнительный анализ языков программирования с примерами решения практически полезных задач, проанализировать вопросы структурной разработки программ, организации удобного взаимодействия программы с человеком и т.п.

Вопросы, относящиеся к внедрению ПК в различные области науки, образования, техники и производства, чрезвычайно многообразны и интересны. Ответы на них должна дать жизнь, которая требует коренного повышения уровня средств обработки информации в стране. Предлагаемая читателю книга является одной из первых в этой области (кстати, авторы работали над ее текстом на персональном компьютере). Несомненно публикации о ПК займут достойное место и в рамках серии "Кибернетика – неограниченные возможности и возможные ограничения". Каждый из последующих сборников должен будет содействовать скорейшему превращению ПК в эффективный инструмент решения самых разнообразных задач школьником, студентом, инженером, конструктором, ученым.

Член-корреспондент АН СССР *И.М. Макаров*

ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ НА РАБОТЕ И ДОМА

Г.Б. КОЧЕТКОВ

Принято считать, что компьютерная революция началась более 40 лет назад. Появившиеся в тот период первые ЭВМ – громоздкие, содержащие огромное число вакуумных ламп и механических частей, по современным представлениям медленно работающие – были неэкономичны, очень ненадежны и мало напоминали гладкие пластмассовые ящики современных микрокомпьютеров. И тем не менее именно их появление ознаменовало громадный прорыв в новую область знания. Уже в те годы многие специалисты, размышляя о путях развития компьютеров, подчеркивали огромный потенциал, заложенный в новой технике, который может оказать глубокое влияние на развитие общества в целом.

Длительное время, однако, этот потенциал оставался скрытым от общества стенами вычислительных центров, подступы к которым надежно охраняли специалисты по эксплуатации ЭВМ и программисты. Его влияние на социально-экономические процессы было опосредованным и не таким значительным, как это предсказывалось в прогнозах 50-х годов. Только с появлением микрокомпьютеров различного назначения и различных типов эта техника стала проникать в цехи, лаборатории, школы, магазины и даже на улицы больших и малых городов. В 80-е годы вычислительная техника становится достоянием широких масс, и с этим связан ряд важных особенностей, которые характеризуют современный этап компьютеризации.

Многие сферы общественной жизни радикально изменяются под влиянием микрокомпьютеров, которые помогают нам решать задачи, экономят наше время, расширяют границы нашего воображения. Последствия этого огромны. Некоторые из них – такие, как "безбумажная технология управления" или "заводы без людей" – уже очевидны. Здесь микрокомпьютеры во много раз повышают производительность труда, а в целом ряде случаев сокращают ненужные в новых условиях профессии. Не столь очевидны другие последствия. Пока неясно, например, как персональные компьютеры повлияют на занятость населения, систему образования, существующую систему связи и многие другие области. И главное, остается открытым вопрос о том, как изменятся наши взгляды и пред-

ставления в эпоху массовой компьютеризации. Все эти вопросы только начинают серьезно обсуждаться, и в рамках данной статьи скорее будут поставлены некоторые из них, нежели будут даны исчерпывающие ответы.

История науки и техники свидетельствует, что любые крупные научно-технические нововведения, такие, например, как автомобиль, самолет, радио, телевидение, проходят в своем развитии три этапа. Сначала они существуют в виде принципиальной идеи и демонстрационных образцов, которые доказывают возможность ее реализации. Затем идею подхватывают специалисты и начинают ее освоение, т.е. создание конкретных технических устройств, и только на третьем этапе техническая новинка становится достоянием масс, начинается серийный выпуск изделий, в основе которых лежит новая идея.

Этот же путь прошла в своем развитии и вычислительная техника. Эпоха инженерной реализации счетно-решающих устройств началась во второй половине 40-х годов. До середины 70-х ЭВМ были вотчиной узкого круга специалистов, знавших технику, приемы работы с машиной и методы программирования. Третий этап развития начался около 10 лет назад, когда два американских энтузиаста – Стефан Возняк и Стив Джобс – смастерили компьютер "Эпл" ("Яблоко"), который сделался своеобразным символом персональной вычислительной техники.

В Советском Союзе идеи создания и применения персональных компьютеров получили широкую популярность среди специалистов по вычислительной технике. Проводятся всесоюзные конференции, посвященные персональным компьютерам, выпущены специальные книги и брошюры, организованы многочисленные курсы по повышению компьютерной грамотности, многие научно-технические журналы открыли соответствующие рубрики и т.п.

Переход от этапа специальных применений к массовому использованию новой технической идеи несет с собой ряд важных качественных изменений.

Во-первых, в этот момент обычно наблюдается скачок в развитии соответствующих отраслей промышленности, увеличивается объем производства и быстро улучшается качество нового товара. Экономические стимулы и другие соображения начинают активно продвигать новинку к покупателю, и одновременно усиливаются требования к ней. Соотношение расходов на разработку товаров массового потребления и на исследовательские работы в данной области резко меняется в пользу первых.

Во-вторых, масштабы внедрения нового начинают расти экспоненциально. Обнаруживаются новые, ранее не осознававшиеся свойства продукта и совершенно неожиданные области применения.

В-третьих, меняются экономические условия использования нововведения. Мерилом его полезности перестают быть прямые затраты — основными соображениями становятся удобство, комфортность, сформировавшиеся потребности людей. Так, наряду с общественным транспортом появился автомобиль в личном пользовании; телефон занял место практически в каждой отдельной квартире. Чисто экономические обоснования этому невозможны.

Высокая стоимость, большие габариты, сложность использования непрограммистами были теми факторами, в силу которых вычислительная техника до поры до времени не выходила за рамки профессионального применения и не могла стать предметом массового потребления.

Дешевизна, компактность, "дружелюбие" к потребителю, характерные для персонального компьютера, резко меняют ситуацию. Прежде всего, персональный компьютер становится центром рабочего места специалиста-непрограммиста — ученого и учителя, инженера и экономиста, литератора и администратора. В отличие от прежних ЭВМ он начинает использоваться индивидуально для профессиональных задач. И более обыденные ситуации постепенно насыщаются электроникой: в магазинах компьютеризуются кассовые аппараты и системы управления товарными запасами; в сфере обслуживания — системы резервирования мест в гостиницах, продажи билетов на различные виды транспорта и спортивно-зрелищные мероприятия и т.п.

Персональная ЭВМ как индивидуальное средство обработки информации хорошо дополняет традиционные системы коллективного пользования. Дело в том, что информация и знания накапливаются в обществе в двух формах: индивидуальной и общественной. Так, несмотря на развитую систему общественных библиотек, каждый из нас стремится иметь свою личную библиотеку, книги которой хранят на полях следы самых неожиданных идей, возникших при чтении. И в век компьютеров процесс накопления индивидуального знания, пусть и обусловленного внешним информационным окружением, остается сугубо личным.

Общественное знание — здание, построенное из кирпичиков знаний индивидуальных. Поэтому персональная информатика — необходимая часть информационных процессов общества и создание вычислительных средств для автоматизации личной информационной деятельности не менее важно, чем разработка супермашин и развитие вычислительных центров коллективного пользования.

Приобретение бытового компьютера в личное пользование порождает потребность в сопутствующих товарах и различных услугах — пакетах прикладных программ, периферийном оборудовании, гибких дисках, техническом обслуживании, специальных

видах связи и т.п. Персональный компьютер готов стать своеобразным ядром новой области потребительского спроса и в этом отношении уподобиться таким товарам, как автомобиль, видеоакустическая аппаратура и пр.

Налицо сейчас и такая тенденция, как усиление роли информационных процессов в экономическом и социально-политическом развитии общества. Персональные компьютеры на производстве представляются наиболее подходящим, высокопроизводительным видом оборудования для переработки информации. Для него достаточно легко и быстро находится сфера приложения там, где существуют либо большие потоки данных, либо массивы отдельных фактов и сведений.

В первую очередь это относится к рабочим местам обширной категории так называемых "информационных работников", основные функции которых связаны с обработкой информации, ее хранением, поиском, передачей, использованием и с созданием новой информации. Доля этой категории в общей структуре занятого населения растет во всех промышленно развитых странах (в США она превысила к началу 80-х годов 50%).

Сегодня одна из центральных задач – определить пути проникновения компьютеров в нашу повседневную жизнь. В прогнозах о будущем персональных компьютеров недостаточно полагаться на уже накопленный опыт их применения в профессиональной сфере. Можно упустить из виду новые области применения, которые пока не обнаружились.

В свое время было сделано немало прогнозов относительно влияния на общественную жизнь автомобиля, самолета, телевизора, и прогнозов не всегда успешных. Известная западноевропейская автомобильная фирма "Мерседес-Бенц", пионер в этой области, в свое время решила не увеличивать производство автомобилей на том основании, что каждый автомобиль требовал в те годы профессионального водителя и при сохранении тогдашних темпов автомобилизации в скором времени все человечество должно было бы превратиться в шоферов. Нельзя сказать, что прогноз был неверен: ныне многие имеют и водят автомашину – но вывод был сделан неправильный. Это показывает, насколько рискованно переносить в будущее существующие формы использования нововведений, и одновременно наводит на определенные аналогии с программистской профессией.

Пример излишне оптимистического прогноза можно привести из истории гражданской авиации. Во время второй мировой войны авиация сделала столь мощный скачок в своем развитии, что появились предположения об авиации как основном виде транспорта в недалеком будущем. Прогноз также не оправдался.

Задача нынешнего этапа развития вычислительной техники, как

представляется, состоит в том, чтобы не поддаваться крайностям при поиске возможных направлений внедрения персонального компьютера. Представление о том, что такое современная ЭВМ, сегодня значительно отличается от того, которое существовало в начале "компьютерной эры" и даже в 60-х или 70-х годах. Создатели ЭВМ предназначали свое детище для выполнения сложных расчетов, рутинных вычислений. Уже в 70-е годы было признано, что главная задача ЭВМ – не вычисления, а управление процессами переработки, сбора, хранения, распределения данных. Появление же персональных компьютеров привело к тому, что и это представление устарело.

Пока, впрочем, спектр потребностей, которые можно удовлетворить с помощью индивидуальных компьютеров, окончательно не определился. Это объясняется тем, что, как техническое нововведение, персональный компьютер переживает еще период младенчества, как автомобиль на рубеже XX века или телевидение в первые послевоенные годы.

Освоение компьютеров происходит быстрее, чем автомобиля или радио, но несколько уступает телевидению. Тому есть свои причины. Техническая новинка может стать массовой лишь при условии "понимания" ее широкими слоями населения, которое не должно испытывать неловкости, замешательства от соседства с новыми машинами. Машина не должна заслонять от пользователя его задачу, потребность, а быть, как говорят, "прозрачной", т.е. попросту удобной в эксплуатации. Телефон, как средство коммуникации вытеснивший в начале века телеграф, технически был значительно более сложным, чем его предшественник, но потребитель не имел (да и сейчас не имеет) ни малейшего об этом представления. Он видел перед собой лишь очень простой в использовании аппарат.

По простоте эксплуатации персональные компьютеры – значительный шаг вперед по сравнению с ЭВМ других классов, однако сравнивать его нужно не с этими ЭВМ, а с той техникой, которую он вытесняет из нашей жизни. Упрощает ли компьютеризация формы и методы управления, решение научно-теоретических задач, ведение хозяйственных расчетов и т.п.? Хотя положительный ответ на этот вопрос довольно очевиден, в ближайшие годы в этой области ожидаются новые радикальные изменения, которые существенно облегчат пользование персональными компьютерами. Контуры этого процесса уже намечаются в наши дни.

Одной из составных частей процесса массовой компьютеризации могут стать системы искусственного интеллекта, которые ныне начинают использоваться в виде так называемых экспертных систем. Основные идеи и положения в области искусственного интеллекта сформировались еще до "пришествия" персональных

компьютеров, и на их основе были созданы образцы экспериментальных программ. Но только с появлением ПК программы искусственного интеллекта начали приобретать настоящий "товарный вид", их взаимодействие с пользователем стало намного удобнее, и они могут сделаться общедоступными. Такое взаимовлияние двух направлений – персональных компьютеров и искусственного интеллекта – может дать исключительные результаты.

Крупное технологическое нововведение определенным образом трансформирует социально-экономические отношения и порождает свою собственную инфраструктуру. Автомобиль потребовал создания шоссейных дорог и развитой системы технического обслуживания, вызвал изменения в энергетическом балансе; телефон повлек за собой создание сети авторелейных станций и разветвленных проводных систем связи. Компьютерная инфраструктура должна включить: систему связи, обеспечивающую возможность прямой коммуникации между ЭВМ различных классов; сферу технического обеспечения и обслуживания ЭВМ; широкий рынок ЭВМ массовых моделей и программного обеспечения; специальные издания, газеты, журналы и т.д. Развивать только промышленное производство компьютеров при отставании остальных элементов инфраструктуры – значит превращать компьютер в довольно дорогостоящую забаву и уподоблять его телевизору без антенны или автомобилю без дорог.

Новая техника в начале своего пути служит удовлетворению (более полному и экономичному) уже сложившихся потребностей общества. В дальнейшем, однако, она открывает перед потребителями новые, ранее не известные возможности. Телефон не только пришел на смену телеграфу, но и потеснил бытовую переписку, породил особую культуру телефонного общения. Электричество, которое вначале служило для освещения домов, создало предпосылки для широкого применения электробытовых приборов – холодильников, пылесосов, стиральных машин и т.п., существенно изменивших облик современного жилища, породивших новый стиль жизни. Эти области применения не были известны в момент, когда новшество прокладывало себе дорогу, а обнаружились позже и получили свое настоящее развитие только на этапе его массового внедрения.

Среди элементов, определяющих лицо современного этапа НТР, можно выделить группу, относящуюся к так называемой "новой информационной технологии". Помимо информатики и вычислительной техники это также радио, телевидение, телефон и некоторые другие, включая космическую связь. Компьютеризация в целом и персональные компьютеры в особенности стали своеобразным ядром, позволившим объединить многие информационные процессы, ранее развивавшиеся независимо. Персональным

компьютерам отводится роль интеграторов информационных процессов, различных по своей природе, которые так или иначе связаны с выполнением непроизводственных функций.

В качестве перспективного направления применения ПК можно выделить его использование в системах коммуникаций и для доступа к банкам информации.

Современное общество обслуживается большой группой систем связи и средств массовой информации: почтой, телеграфом, телефоном, прессой, телевидением, радио. Персональный компьютер, вторгнувшись в эту сферу, может внести в нее большие изменения. Так, компьютерная (электронная) почта может привести к заметному ослаблению традиционной почтовой коммуникации и к уменьшению значения всей существующей системы почтовой связи. Уже получила распространение комбинация настольного компьютера и телефона: при этом компьютер не только записывает полученные по телефону сообщения, но и может автоматически ответить на наиболее простые вопросы. Одновременно он служит информационным терминалом, соединяющим пользователя с банками данных и другими пользователями. Не менее известна комбинация телефона, телевизора и компьютера, получившая название видеотекса.

ПК представляет возможность быстрого обращения к различным электронным источникам информации и ее эффективного использования. Многое из того, что ныне доступно потребителю в форме печатных изданий – справочники, статистические сборники, словари, архивы, рекламные объявления, сведения о купле-продаже, жилищном обмене – может храниться в центральных или локальных банках информации и быть доступно владельцам ПК – абонентам банка и связанной с ним сети ЭВМ. В отдельных случаях абоненту разрешено вносить дополнительные данные и проводить корректировку имеющихся. Организация локальных, специализированных банков информации в сочетании с центральными позволит существенно повысить эффективность многих социально-экономических процессов. Например, члены жилищно-строительного кооператива могут создавать копилку информации о состоянии жилищных условий, произведенных ремонтных работах, принимаемых собранием кооператива решениях и пр.

Перевод всех информационных служб на цифровую технологию позволит быстро и без потерь переносить сведения из одной системы в другую, повысит интенсивность процессов коммуникации. Возникнут новые формы обслуживания, предоставляемые локальными сетями ЭВМ (например, медицинское консультирование, заказ товаров, справочные услуги).

Существует немало прогнозов, согласно которым персональные компьютеры уже при жизни нынешнего поколения получат столь

же широкое распространение, как телефон или телевизор. Не следует, конечно, забывать, что компьютеризация – это сложный социальный процесс, связанный со значительными изменениями в образе жизни населения. Он требует серьезных усилий на многих направлениях, включая ликвидацию компьютерной неграмотности, создание компьютерной инфраструктуры, формирование культуры использования новой информационной технологии и др. Но в то же время сегодня не вызывает сомнения тот факт, что массовый компьютер не просто нужен, он жизненно необходим для ускорения всех процессов социально-экономического развития страны.

КАК УСТРОЕН И РАБОТАЕТ ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

В.М. БРЯБРИН

Что представляет собой современный персональный компьютер? Прежде всего следует выделить три класса электронных вычислительных машин, которые по своим внешним показателям и характеру использования могут быть отнесены к категории персональных компьютеров или персональных ЭВМ (ПЭВМ) : 1) настольные персональные компьютеры; 2) портативные персональные компьютеры; 3) профессиональные рабочие станции.

Машины, относящиеся к 1-му классу, представляют сегодня наибольший интерес, поскольку в них оптимально сочетаются приемлемые внешние показатели и гибкие функциональные возможности, необходимые для построения автоматизированных рабочих мест разного назначения. Дальнейшее изложение будет ориентироваться, главным образом на этот класс ПЭВМ.

Портативные персональные компьютеры, которые ориентируются на применение вне обычных рабочих помещений, начинают постепенно приобретать качества, свойственные более мощным настольным персональным компьютерам. Вполне вероятно, что через несколько лет благодаря дальнейшему совершенствованию машины этого класса начнут доминировать на рынке ПЭВМ.

Профессиональные рабочие станции лежат на другом конце рассматриваемого спектра. Они, как и настольные и портативные ПЭВМ, как правило, рассчитаны на работу одного пользователя, но обладают существенно большими функциональными возможностями, за счет чего растет их стоимость, увеличиваются физические размеры, усложняется эксплуатация, так что эти машины иногда даже относят к категории мини-ЭВМ.

КОМПЬЮТЕР НА ПИСЬМЕННОМ СТОЛЕ – ЧТО У НЕГО ВНУТРИ?

Рассмотрим состав и основные характеристики типичного настольного персонального компьютера. Конструктивно он содержит три основных устройства – системный блок, клавиатуру и дисплей (рис. 1). К этим основным компонентам добавляется еще печатающее устройство – принтер, а в состав системного блока входят средства поддержки коммуникаций для связи с допол-

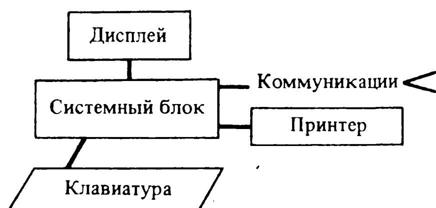


Рис. 1. Базовый комплект настольной ПЭВМ

нительными устройствами, не относящимися непосредственно к основному комплекту ПЭВМ. В таком составе персональный компьютер уже становится пригодным для решения многих задач, свойственных большим и малым ЭВМ.

Системный блок содержит всю электронную начинку ПЭВМ и специальные устройства для постоянного хранения информации на внешних магнитных носителях – накопители на гибких магнитных дисках (НГМД) и на жестких несъемных дисках (НМД).

В состав системного блока входит также блок питания, который преобразует стандартное напряжение электрической сети – 220 или 127 вольт – в напряжение постоянного тока, необходимое для питания электронных компонентов.

Системный блок обычно размещается в небольшом металлическом или пластмассовом корпусе примерно такого же размера, как обычный проигрыватель или магнитофон. Электронные модули размещаются внутри системного блока на отдельных печатных платах. Одна из них – базовая (ее иногда называют генплатой) – содержит основные электронные компоненты и специальные разъемы, в которые устанавливаются дополнительные платы – адаптеры внешних устройств, расширение оперативной памяти и др.

Основой машины является микропроцессор – небольшая по размерам кремниевая пластинка, заключенная в корпус с несколькими десятками выводов. В ней сосредоточена сложнейшая логическая схема, которая является "сердцем" машины. Это так называемое арифметико-логическое устройство и устройство управления, в функции которых входит пересылка отдельных команд и данных между внутренними регистрами, управление ходом вычислений, выполнение самих вычислений, управление взаимодействием внешних устройств микропроцессора. Регистры являются главными носителями информации внутри микропроцессора. Число разрядов в каждом регистре (обычно кратное 8) определяет производительность микропроцессора и машины в целом. Микропроцессоры и построенные на их основе машины называют 8-, 16- или 32-разрядными в соответствии с разрядностью внутренних регистров.

Однако имеет значение не только разрядность внутренних регистров, но и число разрядов внешней шины данных, т.е. число линий, по которым микропроцессор обменивается данными со своими внешними устройствами. Разрядность внешней шины может отличаться от разрядности внутренних регистров. Так, например, во многих персональных компьютерах широко применяются микропроцессоры типа 8086 и 8088. С функциональной точки зрения они совершенно аналогичны, за исключением того, что шина данных у 8086 16-разрядная, а у 8088 – 8-разрядная. При более широкой 16-разрядной шине обмен информацией между микропроцессором и другими устройствами происходит быстрее, но за это приходится платить большим количеством обслуживающих электронных схем, что делает машину более дорогой, энергоемкой и громоздкой. Именно поэтому для дешевых машин предпочитают использовать микропроцессор 8088.

Кроме разрядности регистров, микропроцессор характеризуется максимальным объемом адресуемой памяти. Для 8-разрядных микропроцессоров максимальный объем памяти обычно равен 64 Кбайтам, для 16-разрядных – от 1 до 16 Мбайт. (Термины "Кбайт", "Мбайт" и некоторые другие разъясняются в кратком словаре терминов, приведенном в конце книги). Однако не все адресное пространство может быть фактически доступно для программ. Часть памяти резервируется для системных целей, часть остается просто не занятой. Рабочую память называют оперативным запоминающим устройством (ОЗУ). Помимо ОЗУ, часть адресного пространства занимает постоянное запоминающее устройство (ПЗУ), предназначенное для хранения наиболее важных системных программ и данных. Информация в ПЗУ сохраняется и после выключения машины, благодаря чему не требуется производить ее загрузку с внешних носителей.

Еще один важный параметр микропроцессора – тактовая частота, определяющая скорость выполнения элементарных операций. Тактовая частота задается специальным тактовым генератором. Обычно современные 8- и 16-разрядные микропроцессоры работают с тактовой частотой в диапазоне от 1 до 12 МГц. При этом важно иметь в виду, что для выполнения одной простой операции, например сложения двух целых чисел, взятых из памяти, с пересылкой результата в память – требуется 25–30 тактов. Время выполнения этой операции при средней тактовой частоте около 5 МГц составит 5–6 мкс. Следовательно, микропроцессор может выполнить в этом случае 150–200 тыс. оп./с. При умножении целых чисел на каждую операцию требуется уже 130–150 тактов, поэтому скорость окажется в 5 раз меньше, т.е. составит всего 30–40 тыс. оп./с. При работе с действительными числами, представляемыми в машине с помощью мантиссы и порядка, скорость

2–92

Таблица 1
Основные типы микропроцессоров

Тип	Разрядность основных регистров	Разрядность шины данных	Максимальная адресуемая память (Мбайт)	Тактовая частота (Мгц)
6502	8	8	0,064	1–3
К580	8	8	0,064	3–6
8080	8	8	0,064	3–6
8088	16	8	1	4–8
8086	16	16	1	4–8
80286	16	16	16	6–12
68008	16	8	16	4–6
68000	16	16	16	4–10
68020	32	16	16	4–12
16016	16	16	16	6–14
16032	32	16	16	8–14

упадет еще в 5–8 раз, т.е. составит всего 4–6 тыс. оп./с. Однако здесь могут помочь сопроцессоры – микросхемы, специально сконструированные для ускорения арифметических операций. Арифметические сопроцессоры не могут работать сами по себе, – они всегда действуют в паре с основным микропроцессором. Так, для микропроцессоров 8086 и 8088 имеется арифметический сопроцессор 8087, который позволяет ускорить выполнение операций с действительными числами в 5–10 раз, т.е. поднять скорость до 20–60 тыс. оп./с. Сопроцессор позволяет увеличить и точность арифметических операций, так как обеспечивает работу с числами от 0 до 10 в степени 4098.

С функциональной точки зрения микропроцессор характеризуется системой команд, определяющей порядок программирования всех операций – способов задания операндов, видов адресации и др. При наличии в системе сопроцессора некоторые команды автоматически перехватываются и передаются на исполнение сопроцессору, так что основная программа "не замечает", где в действительности исполняются ее команды.

В настоящее время получили широкое распространение несколько типов базовых микропроцессоров, применяемых для создания ПЭВМ (табл. 1). Наиболее широко используются для построения профессиональных ПЭВМ микропроцессоры семейства 8088/8086, в меньшей степени – микропроцессоры семейства 68000.

Микропроцессор связывается с другими устройствами системного блока через общие линии (проводники), образующие внутрисистемный интерфейс или "общую шину". На

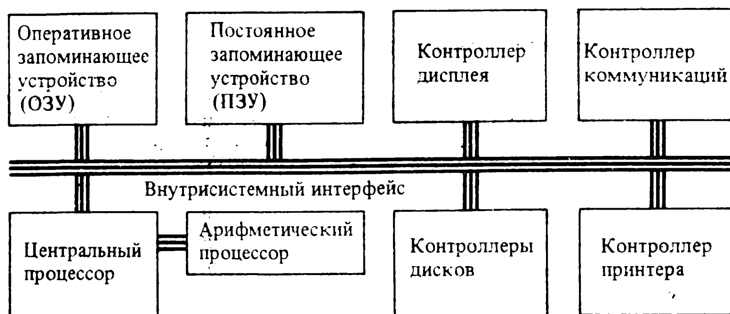


Рис. 2. Внутренняя архитектура системного блока

рис. 2 представлена схема подключения основных компонентов к внутрисистемному интерфейсу. Электронные модули, предназначенные для реализации какой-либо специальной функции (например, для управления накопителями, дисплеем, принтером), называются контроллерами. Они часто реализуются на отдельных печатных платах, вставляемых внутрь системного блока. Такие платы называют адаптерами внешних устройств.

Часто на отдельной плате размещается и дополнительная оперативная память. На базовой плате устанавливается лишь некоторый минимально необходимый объем: 64, 128 или 256 Кбайт, а остальная память выносится на дополнительную плату, так что пользователь может по своему усмотрению регулировать ее объем. Каждая микросхема памяти может содержать 64 Кбита или 8 Кбайт информации. Схема управления памятью организована так, что общий объем памяти должен быть кратен 64 Кбайтам. Следовательно, на один такой "банк" памяти нужно 8 микросхем, к которым обычно добавляется еще одна микросхема для целей автоконтроля. Таким образом, соотношение между объемом памяти и числом требующихся для этого микросхем следующее:

Объем памяти (Кбайт):	64	128	256	384	512	640
Число микросхем:	9	18	36	54	72	90

ГДЕ ХРАНИТСЯ ИНФОРМАЦИЯ, КОГДА КОМПЬЮТЕР ВЫКЛЮЧЕН?

Для постоянного хранения информации в ПЭВМ используются н а к о п и т е л и на гибких и жестких магнитных дисках – НГМД и НМД. В качестве носителей информации для НГМД используются небольшие гибкие диски с нанесенным магнитным покрытием, которые заключены в специальный конверт. Диск вставляется в машину перед началом работы примерно так же, как кассета в магнитофон. Существует несколько типов НГМД, различающихся физическими размерами, плотностью и объемами хранимой ин-

формации и скоростью чтения/записи. В таблице 2 представлены обобщенные параметры основных типов НГМД.

Диски диаметром 203 мм (8 дюймов) используются на мини-ЭВМ, физические размеры которых превышают персональные компьютеры. Однако для ПЭВМ они оказываются чрезмерно велики, поэтому в настоящее время используются лишь в отдельных моделях или как дополнительные накопители для достижения совместимости с мини-ЭВМ. Наибольшее распространение на ПЭВМ получили накопители с диаметром диска 133 мм (5,25 дюйма). Они дешевы, имеют сравнительно небольшие размеры (150 X X 205 X 86 или 150 X 205 X 43 мм) и в то же время позволяют хранить объемы информации, вполне достаточные для большинства применений ПЭВМ. Но в последнее время все больше появляется машин с накопителями диаметром 89 мм (3,5 дюйма). Эти накопители очень компактны (их средние размеры 100 X X 130 X 40 мм) и при этом обладают высокой плотностью записи информации, что позволяет хранить даже большие объемы информации, чем на 133-миллиметровых дисках. Причина заключается в их особом конструктивном исполнении. Микродиски, как их иногда называют, имеют более жесткую конструкцию, что позволяет более точно подводить магнитные головки к поверхности диска. Широкое распространение этого типа накопителей сдерживает лишь тот факт, что бóльшая часть программного обеспечения для ПЭВМ распространяется пока на 133-миллиметровых дисках.

Накопители на жестких несъемных дисках обладают гораздо более высокими показателями благодаря тому, что основной носитель – металлический диск – вращается в замкнутой атмосфере и магнитные головки располагаются очень близко к его поверхности, позволяя увеличивать скорость вращения и обеспечивая очень высокую плотность записи информации. Эти устройства называют иногда дисками типа "Винчестер". Размеры накопителей очень небольшие – не более 90 X 150 X 170 мм. НМД обычно надежнее НГМД, так как они защищены от пыли и имеют лучшее качество головок и магнитных покрытий. Но стоимость их в 5–10 раз превышает стоимость НГМД.

В порядке экспериментов начинают использоваться сменные НМД диаметром 99 мм. Эти устройства по внешним показателям похожи на обычные НМД типа "Винчестер", но в них диски, заключенные в специальные кассеты, могут выниматься из накопителя, что создает безусловные удобства для работы.

Следует сказать несколько слов и о лазерных видеодисках. Эти устройства при небольших размерах (диаметр видеодисков от 5 до 30 см) позволяют хранить огромные объемы информации – от 50 до 4 тыс Мбайт. Они достаточно дешевы и имеют небольшие физические размеры – как обычные НГМД. Главный

Таблица 2
Основные типы накопителей на гибких и жестких магнитных дисках

Тип накопителя	Диаметр диска (мм)	Объем хранимой информации (Мбайт)	Средняя скорость чт./зап. (Кбит/с)
НГМД	203	0,5–1,6	800
НГМД	133	0,2–1,0	250
НГМД	89	0,5–1,0	250
НМД	133	5–20	5000

недостаток этих устройств – сложность оперативной записи информации; только теперь начинают появляться видеодиски с возможностью оперативной записи информации. Уже в ближайшем будущем эти устройства несомненно будут широко использоваться в ПЭВМ для хранения общедоступной информации, системного программного обеспечения, баз данных и др.

С точки зрения операционной системы накопители на дисках являются устройствами прямого доступа. Это означает, что программа может с помощью аппаратных средств прочитать или записать информацию, начиная с любого места диска, – при этом меняется только время подвода головки к нужному сектору и тракту диска. В противоположность этому большинство других внешних устройств – клавиатура, дисплей, принтер, коммуникации – являются устройствами последовательного доступа; работа с ними осуществляется по принципу последовательного чтения и записи.

КАК ИНФОРМАЦИЯ ПОПАДАЕТ ВНУТРЬ ПЭВМ?

Главным средством ввода информации в ПЭВМ является клавиатура. Хорошая клавиатура имеет несколько групп клавиш: алфавитно-цифровые – для ввода чисел и текстов; функциональные – для переключения с одного вида работы на другой; клавиши со стрелками – для перемещения курсора по экрану дисплея; специальные управляющие клавиши – для смены регистров и режимов ввода (рис. 3). Клавиатура изготавливается так, чтобы удовлетворять эргономическим требованиям: она должна быть удобной для длительной работы; расположение клавиш должно соответствовать стандартам, учитывающим нормальные навыки работы на пишущих машинках. Типичные размеры блока клавиатуры 40 X 450 X 180 мм. К системному блоку она подключается с помощью кабеля.

Следует обратить внимание на одну особенность клавиатур современных ПЭВМ. Разработчики прикладных систем всегда

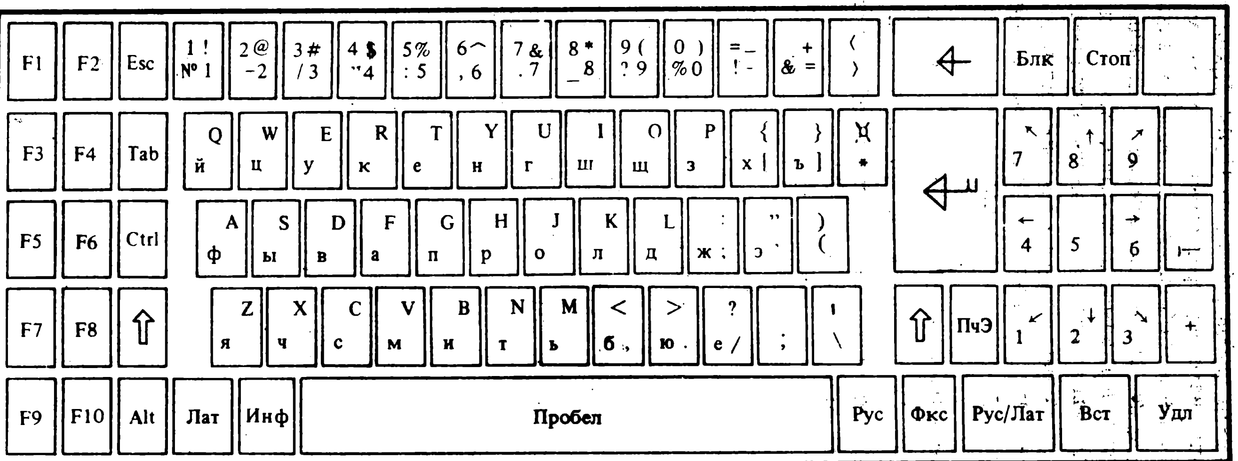


Рис. 3. Клавиатура персонального компьютера

стремятся предельно сократить число требующихся нажатий клавиш при выполнении любой работы на ПЭВМ. Проще всего этого можно достигнуть, изменяя программным путем смысл отдельных клавиш. В отличие от обычных терминалов клавиатура ПЭВМ посылает микропроцессору не код символа, а порядковый номер нажатой клавиши и длительность нажатия. Вся остальная работа по интерпретации смысла нажатой клавиши выполняется программным путем. При таком подходе кодировка клавиш становится независимой от кодировки символов, что резко упрощает работу с клавиатурой и в то же время придает ей большую гибкость.

ДИСПЛЕЙ – ЛИЦО ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Дисплей или монитор служит основным устройством для отображения информации, выводимой во время работы программ. Дисплеи могут различаться весьма существенно, и от их характеристик зависят возможности машин и используемого на них программного обеспечения. Прежде всего, следует различать мониторы, пригодные для вывода только алфавитно-цифровой информации, и графические дисплеи. Другой важный признак – возможность поддержки цветного или только монохромного (черно-белого) изображения. Существенными техническими параметрами являются текстовый формат и разрешающая способность изображения. Текстовый формат характеризуется максимальным числом символов в строке и числом текстовых строк на экране. В графическом режиме разрешающая способность задается числом точек по горизонтали и числом точек (точечных строк) по вертикали. Другой характерный параметр – это количество поддерживаемых уровней яркости в монохромном режиме или количество цветов при цветном изображении. Наконец, немаловажным параметром является размер экрана, от которого зависит различимость изображения в целом и четкость отдельных элементов, в том числе букв и цифр.

Указанные параметры зависят как от конструкции экрана, так и от схемы управления, сосредоточенной в системном блоке. В настоящее время в большинстве случаев применяется схема формирования изображения на основе экранной, или растровой памяти (bit-mapping). Каждый элемент изображения – одна точка на экране дисплея – формируется из фрагмента растровой памяти, состоящего из 1, 2 или 4 бит. Информация в указанных битах управляет яркостью или цветом точки на экране, а также ее миганием и другими возможными атрибутами. При этом объем растровой памяти прямо связан с разрешающей способностью дисплея. Так при 640X200 точках и двух уровнях яркости в монохромном режиме нужна растровая память объемом 16 Кбайт;

Таблица 3

Типичные параметры дисплеев на основе ЭЛТ

Текстовые форматы	25 строк по 80 символов 25 строк по 40 символов
Разрешающая способность в графическом режиме	640 точек по горизонтали 200 точек по вертикали
Количество цветов, выводимых одновременно	от 4 до 16
Количество цветовых гамм (палетт)	от 2 до 256
Размер экрана по диагонали	30,5; 33; 35,6 мм

если же при этом нужно будет управлять 16 цветами каждой точки, то потребуется 64 Кбайта; а при 1024 X 1024 точках и двух цветах потребуется уже 132 Кбайта растровой памяти. Цифры и буквы при таком методе управления выводятся на экран с помощью специальных знакогенераторов – электронных схем, управляемых точечными матрицами (8 X 8, 6 X 8, 8 X 12 или другого формата), на которых формируются изображения букв и цифр.

Большинство профессиональных ПЭВМ имеет дисплеи, основанные на монохромных или цветных электронно-лучевых трубках (ЭЛТ). Управление цветной ЭЛТ осуществляется, как правило, с помощью 5 стандартных сигналов – для красного, зеленого и синего лучей, а также вертикальной и горизонтальной синхронизации. Такие устройства называют RGB-мониторами (произносится "эр-же-бэ-монитор"). Данное обозначение происходит от английских слов red – красный, green – зеленый, blue – синий.

Подключение дисплея к системному блоку осуществляется с помощью контроллера, который чаще всего реализуется в виде отдельной платы (адаптера), вставляемой в системный блок. Адаптер обычно содержит растровую память и схемы управления ЭЛТ. Кроме того, на нем размещается микросхема ПЗУ, в которой записываются образы знаковых матриц, выводимых на экран. Сменив эту микросхему, можно тем самым изменить знакогенератор, т.е. начертания букв, цифр и других знаков, выводимых на экран. Контроллер согласуется с типом дисплея, для которого он предназначен. Одни контроллеры позволяют подключать только монохромные дисплеи, другие – цветные RGB-мониторы, третьи – обычные телевизоры. Контроллеры, допускающие работу только с алфавитно-цифровой информацией, намного проще графических контроллеров; однако в современных ПЭВМ возможность работы с графической информацией абсолютно необходима. Типичные параметры дисплеев на основе ЭЛТ приведены в табл. 3.

Для дешевых домашних или школьных компьютеров часто в качестве дисплея используется обычный бытовой телевизор. Но он имеет ограниченную разрешающую способность – не более 256 X 256 точек, что, конечно, не позволяет получать качественное изображение. Другой тип дисплея – на основе жидкокристальных панелей – используется в портативных ПЭВМ. В некоторых моделях ПЭВМ такие дисплеи позволяют получать почти такую же разрешающую способность изображения, как на ЭЛТ. Но оно, во-первых, не может быть цветным, во-вторых, требует хорошей освещенности и правильного угла зрения. Поэтому пока такие дисплеи не очень распространены.

ВЫВОД ИНФОРМАЦИИ НА БУМАГУ

П р и н т е р ы, хотя и не входят непосредственно в состав ПЭВМ, но являются непременным атрибутом автоматизированных рабочих мест на их основе. Принципы действия современных принтеров для ПЭВМ различны. Главными показателями являются качество печати и скорость (табл. 4).

Различаются следующие условные разновидности качества печати: черновая, усиленная, почти типографская и типографская. В табл. 4 и в данном тексте эти разновидности качества названы соответственно посредственным, средним, хорошим и отличным. Скорость печати у разных принтеров колеблется от 12 символов/с до 8 страниц/мин (около 300 символов/с).

Наиболее часто с ПЭВМ используются м а т р и ч н ы е принтеры. В них изображения отдельных знаков строятся на матрице размером 9 X 9, точки которой формируются ударами через красящую ленту тончайших стержней (диаметром 0,2–0,3 мм). Число стержней обычно равно 9, так что точки в пределах матрицы соприкасаются, образуя непрерывные линии. Печатающий узел – головка со стержнями – двигается в горизонтальном направлении, и знаки могут печататься как при прямом, так и при обратном ходе, что позволяет достигать высокой скорости печати – до 200 знаков в секунду.

Качество печати у принтеров этого типа от посредственного до хорошего, но зато они дешевы, компактны, не требуют специальной бумаги и работают довольно быстро. Кроме того, эти принтеры обладают важным достоинством: поскольку движением стержней, шагом перемещения головки по горизонтали и движением бумаги по вертикали можно управлять с помощью программы, то легко создавать произвольные шрифты, а также выводить на принтер любые графические изображения. Некоторые принтеры заранее снабжаются ПЗУ с несколькими шрифтами; в других принтерах имеется буфер, в который можно загрузить в начале работы

Таблица 4

Основные типы принтеров для ПЭВМ

Тип принтера	Качество печати	Разрешающая способность (точ./мм)	Скорость печати (зн./с)
Матричный	Посредственное	2–3	90–200
	Среднее	3–4	30–90
	Хорошее	4–6	30–80
С шрифтоносителем	Отличное	–	12–60
Термографический	Среднее	3–5	100–200
	Хорошее	5–7	60–180
Струйный	Среднее	3–4	40–150
	Хорошее	4–5	20–80
Лазерный	Отличное	10–12	200–300

свой шрифт, и затем использовать его наряду со встроенным стандартным шрифтом. Большинство точечно-матричных принтеров имеет также разные режимы печати и несколько размеров шрифтов. Некоторые высококачественные принтеры имеют головки с 24 стерженьками, что позволяет печатать знаки почти типографского качества.

Другой тип принтеров основан на использовании *сменных шрифтоносителей* в виде дисков с нанесенными литерами какого-либо алфавита. Эти принтеры, называемые иногда ромашковыми, дают отличную печать типографского качества (подобно хорошей пишущей машинке), но их очевидные недостатки – низкая скорость (до 60 знаков/с) и возможность использования лишь стандартных шрифтов, имеющихся на шрифтоносителях.

Принтеры *термографического* типа компактны, дешевы, бесшумны и могут давать среднее и хорошее качество печати, но они требуют специальной бумаги, не всегда имеющейся в распоряжении пользователей. Некоторые из них воздействуют теплом прямо на бумагу, другие растапливают красящий состав, который затем ложится на бумагу. Эти принтеры менее долговечны по сравнению с другими типами. Чаще всего они применяются в портативных ЭВМ ввиду малых размеров.

Струйные принтеры привлекли в последнее время внимание тем, что они работают бесшумно, могут давать среднее и хорошее качество печати, в том числе с несколькими уровнями яркости, а некоторые из них обеспечивают и цветную печать. Принцип действия основан на том, что под управлением программы из перемещающегося по горизонтали сопла на бумагу выбрасываются мель-

чайшие капельки чернил, формируя необходимые изображения. Однако пока струйные принтеры не особенно распространены; их главный недостаток – необходимость специальных чернил, не засоряющих сопло головки, и чувствительность к качеству бумаги. Эти принтеры не позволяют делать сразу несколько копий.

Последнее слово техники печатающих устройств связано с л а з е р н ы м и принтерами. Улучшение конструкции лазера, оптической части и печатающего узла привело к значительному удешевлению лазерных принтеров, уменьшению их габаритов и увеличению надежности. Лазерный луч сканирует под управлением программы по поверхности барабана, с которого затем на бумагу переносится красящее вещество. Качество печати у лазерных принтеров отличное, скорость работы – до 8 строк/мин. В графическом режиме эти принтеры печатают с разрешающей способностью около 12 точек/мм. Возможно использование встроенных или загружаемых шрифтов. По габаритам лазерные принтеры сравнимы с телевизором среднего размера. Но пока они остаются дороже других типов принтеров. Кроме того, довольно дорог сменный печатающий узел, включающий барабан с красящим веществом.

Все типы принтеров присоединяются к ПЭВМ через посредство специального контроллера, реализующего протокол обмена на основе параллельного или последовательного интерфейса.

ЧТО ЕЩЕ МОЖНО ПОДКЛЮЧИТЬ К ПЕРСОНАЛЬНОМУ КОМПЬЮТЕРУ?

Из множества различных устройств, часто подключаемых к ПЭВМ, следует упомянуть особый манипулятор типа "м ы ш ь" ("колобок"). Это небольшая коробочка, скользящая по плоской поверхности; относительные координаты ее перемещения передаются в машину и обрабатываются таким образом, чтобы управлять движением на экране дисплея специально выделенного маркера, который называют курсором. Такой способ выбора позиции и указания объектов на экране очень удобен при работе с изображениями.

Для ввода в ПЭВМ карт и чертежей используются специальные графические планшеты – д и д ж и т а й з е р ы. На планшет накладывается лист бумаги с изображением, которое нужно в закодированном виде ввести в машину. Пользователь перемещает по планшету специальный указатель, нажимая кнопку в местах соединения прямых отрезков; при этом координаты концов отрезков вводятся в машину, образуя необходимую структуру данных.

Для вывода чертежей, графиков и других изображений на бумажные листы (их называют в этом случае твердой копией),

используются компактные графопостроители (плоттеры), специально изготавливаемые для совместной работы с персональными компьютерами.

Все эти устройства, как правило, подключаются к системному блоку с помощью коммуникационных адаптеров, которые обеспечивают передачу сигналов в соответствии со специальными соглашениями об их электрических и временных параметрах. Такие соглашения называются протоколами. Чаще всего применяются стандартные последовательные или параллельные протоколы RS-232C или Centronix. Коммуникационные адаптеры обеспечивают также соединение ПЭВМ с другими ЭВМ, с локальной сетью, с приборами, станками и другими устройствами.

Большое распространение имеют так называемые многофункциональные платы для ПЭВМ. Дело в том, что для реализации некоторых функций, например подключения принтера или коммуникационных каналов, требуется совсем немного микросхем. Занимать под отдельные контроллеры целые платы невыгодно, – получается слишком много плат с небольшим заполнением. Поэтому на одной плате часто объединяют контроллеры для выполнения нескольких функций, а также размещают дополнительную оперативную память. Например, получили распространение многофункциональные платы, на которых размещаются: 384 Кбайта оперативной памяти; контроллер принтера; контроллер последовательного коммуникационного канала; контроллер графического манипулятора; встроенный календарь и часы с батарейкой.

Существуют платы и с другими типами контроллеров. Подбирая различные сочетания многофункциональных плат, владелец ПЭВМ может создать для себя наиболее удобную конфигурацию машины.

При включении ПЭВМ в локальные сети они начинают играть роль отдельных рабочих станций в большой информационной системе. При этом появляется возможность совместного использования дорогих устройств ввода-вывода, например дисков большого объема, лазерных принтеров и др. Локальная сеть позволяет отдельным рабочим станциям обмениваться информацией друг с другом; передаваемые по сети данные могут нести некоторую приватную или служебную корреспонденцию (электронная почта) или это могут быть программы и данные, необходимые для совместной работы нескольких пользователей. При подключении к такой сети больших ЭВМ персональные компьютеры могут выступать в роли "интеллектуальных терминалов", обеспечивающих значительно больше возможностей для работы, чем обычные алфавитно-цифровые терминалы. Соединение отдельных узлов в сети может базироваться на разных архитекту-

Таблица 5

Параметры устройств, входящих в состав 16-разрядного персонального компьютера

Тип основного микропроцессора	К1810ВМ8086
Тактовая частота	5 Мгц
Максимальная адресуемая память	1 Мбайт
Максимальный доступный объем ОЗУ	640 Кбайт
Объем внешней памяти одного НГМД	360 Кбайт
Объем внешней памяти НМД "Винчестер"	10 Мбайт
Разрешающая способность дисплея в графическом режиме (количество точечных элементов изображения по горизонтали и вертикали)	640 × 400 640 × 200 320 × 200
Формат представления на дисплее алфавитно-цифровой информации (число строк и знаков в строке)	80 × 25 40 × 25
Скорость работы принтера (зн/с)	от 30 до 300

рах – шинной, кольцевой, звездообразной, кластерной (в виде отдельных пучков) и их комбинациях. Основной параметр сети – ее пропускная способность, которая может лежать в диапазоне от нескольких сот Кбайт/с до 10 Мбайт/с.

Описанные технические характеристики свойственны многим популярным персональным компьютерам. В табл. 5 приведены типичные параметры устройств, применяемых в 16-разрядных ПЭВМ.

ПОРТАТИВНАЯ ПЭВМ – МАШИНА ДЛЯ ДОМА, ШКОЛЫ И УЧРЕЖДЕНИЯ

Одновременно с использованием настольных моделей ПЭВМ получают распространение так называемые портативные ЭВМ. Этот класс машин возник на базе очень простых дешевых микрокомпьютеров, ориентированных на домашнее применение. Такие машины обычно собираются на одной печатной плате, установленной внутри небольшого плоского корпуса размером с книгу. В этот же корпус встраивается простая клавиатура; кроме того, он имеет разъемы для подключения бытового телевизора и магнитофона. В основе такой машины лежит обычно 8-разрядный микропроцессор, оперативная память имеет объем от 16 до 64 Кбайт, а в состав программного обеспечения входит лишь язык Бейсик. В качестве внешнего накопителя у таких машин используется стандартный бытовой магнитофон, на кассету которого можно записать несколько сот Кбайт информации. Чаще всего на кассетах хранятся игровые программы. Роль дисплея играет обычный цветной телевизор, на антенный вход которого

через встроенный контроллер подается соответствующий сигнал. Иногда такие машины снабжаются специальными мониторами, которые дают лучшее качество изображения, чем простые телевизоры.

Развитие школьной информатики привело к некоторой модификации дешевых компьютеров, которые первоначально предназначались лишь для игр с использованием домашних телевизоров. Появилась необходимость подключения к таким ПЭВМ не только бытовых магнитофонов, но и накопителей на гибких магнитных дисках, которые могли бы стоять рядом с машиной. При этом, кроме языка Бейсик, появляется возможность использования других программных систем, например языка Лого, операционной системы CP/M, а вместе с ней – множества других программ. Кроме того, потребовалось объединять несколько машин в небольшую "кабинетную" сеть; с этой целью в системный блок встраиваются специальные контроллеры. Было выдвинуто предложение о принятии стандарта на школьные ЭВМ. Этот стандарт, получивший название MSX, предусматривает следующие параметры школьной машины:

микропроцессор – 8-разрядный типа К580;

объем ПЗУ (для Бейсика) – 32 Кбайта;

объем ОЗУ – от 8 до 64 Кбайт;

текстовый формат дисплея – 24 строки по 32 знака;

разрешающая способность дисплея в графическом режиме – 256 X 192 точки, 16 цветов;

внешние накопители – на основе гибких дисков диаметром 133,89 и 76 мм.

возможность выдачи звука – на 8 октавах по 3 каналам;

программное обеспечение – язык Бейсик и операционная система MSX.

Новое поколение портативных персональных компьютеров ориентируется уже на применение в профессиональной деятельности. Одно из новшеств заключается в использовании вместо бытового телевизора встроенного жидкокристального дисплея большого размера, который позволяет выводить как тексты (от 8 до 25 строк по 80 символов), так и графические изображения с разрешающей способностью до 640 X 256 точек. Жидкокристальный дисплей, хотя и не очень удобен для работы, но зато очень компактен.

Следующее усовершенствование заключается во введении встроенного накопителя на основе миниатюрных флоппи-дисков или микрокассет. Это позволяет при сохранении очень небольших размеров машины иметь в ее составе практически все необходимые для профессиональной работы устройства. Даже компактные термографические принтеры стали встраиваться в системный блок, хотя чаще они подключаются извне.

Наконец, главные изменения произошли в связи с переходом в портативных машинах на 16-разрядные микропроцессоры — те же, что используются в настольных моделях. Это позволяет создавать портативные ЭВМ, почти полностью совместимые с популярными моделями настольных ПЭВМ: иметь на них большую оперативную память, использовать одни и те же операционные системы, языки программирования и прикладные системы. Такая машина позволяет человеку работать дома или в другом месте вне своего учреждения, а затем с помощью коммуникационного адаптера подсоединить ее к настольной ПЭВМ и перенести результаты работы на более мощный настольный компьютер.

Конструктивно портативная ЭВМ выполняется так, чтобы в сложенном виде легко разместиться в небольшом чемоданчике или портфеле; в рабочем состоянии она раскрывается подобно книге — одна половина, лежащая на столе, содержит клавиатуру, вторая половина, содержащая дисплей, наклоняется к первой под углом. Принтер может быть установлен рядом. Машина, как и принтер, может питаться от сети или от встроенных аккумуляторных батарей. Ее вес 3–4 кг.

Таким образом, при внешних атрибутах почти "карманной" машины такая портативная ЭВМ обладает свойствами, позволяющими использовать ее для серьезной профессиональной работы.

ПРОФЕССИОНАЛЬНЫЕ РАБОЧИЕ СТАНЦИИ

Настольные и портативные ПЭВМ широко применяются для автоматизации массовых видов деятельности — подготовки различных документов, моделирования экономических систем, планирования, обучения и др. Имеются, однако, такие сферы деятельности, где возможности этих машин могут оказаться недостаточными. Сюда в первую очередь относятся автоматизация проектирования и сложные научные исследования. Для создания автоматизированных рабочих мест в этих областях применяются более мощные машины, которые называются профессиональными рабочими станциями.

Архитектура этих машин часто основывается не на стандартных микропроцессорах, а на микропрограммной реализации специализированной системы команд. Микропрограммирование — это особый прием, основанный на специальной организации арифметико-логического устройства и управления. Часть электронных схем обеспечивает выполнение самых элементарных операций (микрокоманд), таких, как пересылка информации из регистра в регистр или сложение содержимого двух регистров.

Другая группа схем позволяет задавать комбинации микрокоманд (микропрограммы), которые обеспечивают выполнение более сложных действий, например сложение двух действительных чисел, условный переход или даже вывод точек на экран дисплея. Микропрограммы могут изменяться на стадии проектирования машины. При этом появляется возможность создания машины с различными, проблемно-ориентированными системами команд.

Собственно системное и прикладное программирование осуществляется здесь уже в терминах более крупных макрокоманд и, конечно, является более эффективным, чем при использовании универсального набора команд, предоставляемого стандартным микропроцессором. Поэтому на основе такого подхода могут быть построены машины, ориентированные на эффективную поддержку какого-либо класса задач.

Наиболее известны так называемые Лисп-машины – рабочие станции, базирующиеся на использовании языка Лисп. Этот язык давно известен как идеальное средство для программирования задач искусственного интеллекта – создания "баз знаний", автоматического выполнения логического вывода, создания экспертных систем и др. Однако на обычных ЭВМ со стандартными системами команд программы на Лиспе работают довольно медленно, и объемы памяти в них часто бывают недостаточны для построения практически полезных систем. В Лисп-машинах микропрограммными средствами реализуется система команд, поддерживающая базовые операции языка Лисп. Машинные слова в такой машине гораздо длиннее, чем в обычных 16-разрядных персональных компьютерах – они содержат 36 бит, а 32-разрядный адрес теоретически позволяет иметь память объемом до 1 Гбайта (256 миллионов 4-байтовых слов). Конечно, физическая оперативная память пока не может быть такой большой – это стоило бы слишком дорого. Поэтому организуется так называемая виртуальная память – при этом часть требуемой информации находится в реальной физической памяти (объемом 2–4 Мбайта), а другая, большая часть находится на внешних носителях и читается оттуда по мере необходимости; но основная программа "не замечает" этого механизма и обращается одинаковым образом ко всем адресуемым словам. Большинство операций выполняется с быстродействием около 5 млн оп./с. Виртуальная память и файловая система основана на использовании дисков "Винчестер" большого объема – 80, 170 или 400 Мбайт.

Большое внимание в таких машинах уделяется поддержке работы с растровыми дисплеями высокого разрешения (1100 X 900 точек), что позволяет работать с высококачественны-

ми изображениями. Основной дисплей обычно монохромный, но может подключаться и цветной с такой же разрешающей способностью. Конструктивно системный блок оформляется в виде небольшой стойки размером с тумбочку, к которой с помощью кабелей подключается дисплей и клавиатура. Стоимость таких машин в 5–10 раз выше, чем обычных настольных ПЭВМ, но и предоставляемые ими возможности также весьма велики. Совершенствование средств микроэлектроники и их удешевление, по-видимому, позволит уже в ближайшем будущем существенно уменьшить их размеры, стоимость и приблизить к современным ПЭВМ.

Другой тип рабочих станций, более дешевых и похожих на обычные персональные компьютеры, основывается на стандартных микропроцессорах с 32-разрядными словами. В этих машинах увеличен объем оперативной памяти до 1–4 Мбайт. Тактовая частота 10 МГц обеспечивает высокое быстродействие системы. Растровый монохромный дисплей с большим экраном (47 см по диагонали) имеет разрешающую способность 1100 X 900 точек, а дополнительный цветной дисплей – 640 X 480 точек с возможностью отображения 256 цветов из 16 миллионов наборов. Благодаря этому обеспечиваются богатые возможности работы с графическими изображениями, шрифтами разной формы и размера, многооконным отображением информации, двух- и трехмерными графическими объектами. Внешний накопитель может базироваться на жестких дисках объемом от 40 до 200 Мбайт. Предусматривается установка магнитных лент для сброса информации с дисков. Машина снабжается аппаратными и программными средствами для подключения к локальной сети.

При относительно невысокой стоимости (в 2–4 раза выше, чем стоимость массовых ПЭВМ) такая машина с успехом может служить для организации автоматизированных рабочих мест в различных областях, включая проектирование электронных и машиностроительных изделий, медико-биологические исследования, геофизику и другие области.

ОПЕРАЦИОННАЯ СИСТЕМА – "ПРОГРАММНАЯ ОБОЛОЧКА" АППАРАТНЫХ СРЕДСТВ ПЭВМ

Необходимой составной частью любой ЭВМ является ее п р о г р а м м н о е о б е с п е ч е н и е (ПО). Без соответствующих программ практически невозможно заставить машину сделать что-либо полезное. Для массового внедрения персональных компьютеров необходимо программное обеспечение, ориентированное как на универсальное применение, так и на отдельные проблемные области. В настоящее время для разных типов ПЭВМ

Типы микропроцессоров

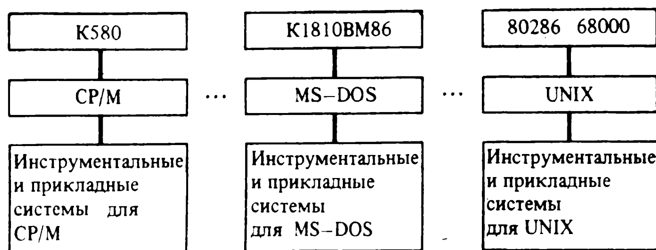


Рис. 4. Основные группы программного обеспечения для ПЭВМ

разработано несколько десятков тысяч программ, которые могут быть разделены на следующие основные классы:

1. Операционные системы.
2. Инструментальные языки и системы программирования.
3. Прикладные системы.

Операционная система (ОС) является неотъемлемой частью компьютера, обеспечивающей управление всеми аппаратными компонентами и позволяющей отделить остальные классы программ от непосредственного взаимодействия с аппаратурой. Число типов ОС невелико – не более нескольких десятков, но их роль чрезвычайно важна.

Прежде чем переходить к рассмотрению основных классов ОС, следует обратить внимание на существование ЭВМ, в которых они носят вырожденный характер. В дешевых домашних или школьных компьютерах, поддерживающих лишь язык Бейсик и игры, функции ОС обычно погружены "внутрь " Бейсика. Это значит, что сразу после включения машины пользователь попадает в операционную среду языка Бейсик. При этом он может вводить с клавиатуры текст программы или набрать команду, обеспечивающую запуск программы, распечатку ее текста, обращение к диску или магнитофону для записи/чтения программы и др. Все команды расшифровываются и выполняются самим интерпретатором языка Бейсик. Таким образом, ОС и Бейсик представляют на таких машинах единое целое. Некоторые модели дешевых машин ориентируются не на Бейсик, а на другой базовый язык – Лого или Форт; однако все приведенные выше рассуждения остаются справедливыми и в этих случаях.

На более мощных персональных компьютерах, где необходимо поддерживать работу разных языков программирования и прикладных программ, без отдельной ОС уже нельзя обойтись.

В настоящее время получили широкое распространение и фактически стандартизовались несколько "семейств" ОС, ориенти-

рованных на определенные типы микропроцессоров. Наиболее распространены следующие типы ОС – CP/M, MS-DOS и UNIX, которые мы еще рассмотрим подробнее. Из этих систем первая используется на машинах с 8-разрядными, вторая – с 16-разрядными, третья – преимущественно с 32-разрядными микропроцессорами.

Все остальное программное обеспечение можно разделить на большие группы, связанные с соответствующими семействами ОС. Программы, созданные для работы с определенным классом ОС, обычно не работают под управлением других систем. Следовательно, прикладное программное обеспечение, а также инструментальные языки и системы программирования для ПЭВМ можно разделить на несколько больших групп, обусловленных существованием для каждого типа микропроцессора нескольких операционных систем (рис. 4).

ДЛЯ ЧЕГО НУЖНА ОПЕРАЦИОННАЯ СИСТЕМА?

ОС обеспечивает выполнение двух главных задач:

1) поддержку работы всех программ и обеспечение их взаимодействия с аппаратурой;

2) предоставление пользователям возможности общего управления машиной.

В рамках первой задачи ОС обеспечивает взаимодействие программ с внешними устройствами и друг с другом, распределение оперативной памяти, выявление различных событий, возникающих в процессе работы, и соответствующее реагирование на них (например, при ошибочных ситуациях) и др. Общее управление машиной осуществляется на основе командного языка, с помощью которого человек может осуществлять такие операции, как разметка дисков, копирование файлов, распечатка каталогов на экране дисплея, запуск любых программ, установка режимов работы дисплея, принтера, коммуникаций и другие действия.

В различных моделях ПЭВМ используются ОС с разной архитектурой и возможностями; для их хранения и работы необходимы различные ресурсы оперативной памяти; они предоставляют пользователям разную степень сервиса для разработки программ и их использования.

Рассмотрим наиболее простую ОС, предоставляющую пользователям лишь самый необходимый набор средств для управления ресурсами ПЭВМ, доступа к файловой системе и организации диалога. Такой "минимальный" подход реализуется в однопользовательских ОС, которые применяются для большинства 8-разрядных ПЭВМ. Создание удобного (дружественного) интерфейса, поддержка специфических внешних устройств, реа-

лизация общих сервисных функций возлагается в этом случае не столько на ОС, сколько на прикладные программы, работающие под их управлением. ОС данного класса дают примерно одинаковые возможности как системному программисту, так и конечному пользователю, поэтому их применение оправдано для дешевых персональных компьютеров, часто попадающих в руки программистов-любителей, – людей, начинающих заниматься программированием из любопытства и очень быстро понимающих, что ПЭВМ могут принести реальную пользу в их профессиональной деятельности. Наибольшее распространение среди систем такого типа получили ОС семейства CP/M.

Другой класс образуют однопользовательские ОС с более развитыми средствами доступа ко всем аппаратным компонентам, гибкой файловой системой, основанной на иерархической структуре каталогов, удобным для пользователя командным языком. Средства, предоставляемые ОС этого класса, позволяют, с одной стороны, формировать удобную операционную обстановку для разработки программного обеспечения; с другой стороны, на их основе довольно легко можно создавать автоматизированные рабочие места с простыми средствами доступа конечных пользователей к прикладным пакетам и программам. К этому классу относятся ОС семейства MS-DOS, получившие широкое распространение для 16-разрядных персональных компьютеров.

Третий класс ОС ориентирован в основном на эффективную поддержку процесса разработки программного обеспечения. Наиболее яркими представителями этого класса являются ОС семейства UNIX. Здесь, как и в системах второго класса, имеется развитая файловая система, обеспечивается программирование доступа ко всем типам внешних устройств, имеется очень мощный командный язык. Кроме того, в состав системы входит множество сервисных программ ("утилит"), обеспечивающих выполнение разнообразных функций, потребность в которых систематически возникает при разработке программного обеспечения. Наконец, в этих системах заложена возможность организации одновременной работы нескольких терминалов, хотя для фактического использования этой возможности нужна уже более мощная аппаратурная поддержка, что превращает персональный компьютер в довольно дорогую рабочую станцию. Системы этого типа требуют значительных ресурсов, не всегда доступных на дешевых ПЭВМ, в то время как их мощность часто является избыточной с точки зрения большинства так называемых конечных пользователей, занятых не столько разработкой программ, сколько решением своих профессиональных задач.

Наконец, особый класс составляют ОС, ориентированные главным образом на поддержку удобной работы конечных

пользователей. Такие системы имеют развитые средства поддержки диалога с использованием графики, работы с дисплейными окнами, специальных манипуляторов ("мышь", джойстик") для выбора объектов и операций над ними. К сожалению, для разработки программ в рамках таких ОС упомянутые выше особенности мало что дают. Программирование как таковое в ОС этого типа, конечно, тоже доступно с использованием соответствующих трансляторов, однако в этом случае программист не получает никаких преимуществ для своей работы. Наоборот, его работа замедляется из-за необходимости производить несвойственные манипуляции с графическими объектами вместо использования лаконичных текстовых команд, а также из-за того, что довольно значительные ресурсы – оперативная память и время микропроцессора – расходуются на поддержку удобного интерфейса, в котором системный программист не испытывает большой нужды.

Из каких частей состоит ОС? В полной конфигурации ОС для персонального компьютера, ориентированного на профессиональное применение, должна содержать следующие основные компоненты: файловую систему, драйверы внешних устройств, процессор командного языка. Рассмотрим указанные компоненты подробнее.

ФАЙЛОВАЯ СИСТЕМА – ХРАНИЛИЩЕ ПРОГРАММ И ДАННЫХ

Одна из важнейших функций ОС – организация файловой системы. Файл – это место для постоянного хранения информации – программ, данных для их работы, текстов, закодированных изображений и др. Реализуются такие файлы как участки памяти на внешних магнитных носителях – гибких или жестких магнитных дисках. Каждый файл имеет имя, зарегистрированное в каталоге файлов – "директории". Директорий доступен пользователю через командный язык операционной системы: его можно просматривать, переименовывать зарегистрированные в нем файлы, копировать их на новые места и удалять. Директорий сам по себе можно представлять в виде именованного файла и хранить в другом директории; так образуются иерархические файловые структуры. К файловой системе имеет доступ также и любая прикладная программа, для чего во всех языках программирования имеются специальные процедуры. Понятие файла может быть обобщено на любой источник или потребитель информации в машине, например коммуникационный канал, принтер, дисплей, клавиатуру и др. Такая трактовка, принятая в развитых ОС, например в системах MS-DOS и UNIX, создает удобства для организации взаимодействия программ и обмена информацией с внешними устройствами.

Для персональной ЭВМ файловая система в определенной сте-

пени является сердцевиной всего системного программного обеспечения. Структура файловой системы и структура хранения данных на внешних магнитных носителях определяют удобство работы пользователя, скорость доступа к файлам, возможность создания хороших баз данных и т.д. От файловой системы во многом зависит организация многопользовательской работы, если она поддерживается на данной модели ПЭВМ. Таким образом, возможности, предоставляемые файловой системой, накладывают отпечаток на всю работу ОС, а значит и пользователя. В однозадачных ОС файловая система является наиболее крупной составной частью.

КАК УПРАВЛЯЮТ ВНЕШНИМИ УСТРОЙСТВАМИ?

ПЭВМ может иметь довольно большой набор внешних устройств (ВУ). Помимо стандартных ВУ – дисплея, клавиатуры, гибких дисков, жестких дисков и принтера, к машине могут подключаться по последовательным и параллельным коммуникационным каналам дополнительные устройства ввода-вывода: графопостроители, планшеты, манипуляторы типа "мышь", а также специфические устройства – модемы для связи с телефонными линиями, контроллеры локальных сетей, аналого-цифровые и цифроаналоговые преобразователи и другое оборудование. Более того, даже стандартные устройства, например принтеры, могут иметь несколько режимов работы и считаться вследствие этого разными устройствами. Каждое ВУ характеризуется своей пропускной способностью и структурой передаваемых и принимаемых данных.

Поддержка всего этого набора ВУ – одна из важнейших функций ОС. Для осуществления этого введено понятие *драйвера* – программы специального типа, ориентированной на управление внешним устройством. Каждому типу ВУ сопоставляется свой драйвер. Драйверы стандартных устройств образуют в совокупности базовую систему ввода/вывода (БСВВ), которая часто заносится в ПЗУ системного блока. Драйверы дополнительных устройств могут подключаться динамически при запуске машины. Некоторые типы ОС предоставляют средства для составления новых драйверов, ориентированных на особые устройства.

ВЗАИМОДЕЙСТВИЕ ПОЛЬЗОВАТЕЛЯ С ОПЕРАЦИОННОЙ СИСТЕМОЙ

Загрузка готовых программ из файлов в оперативную память ПЭВМ и их запуск по командам пользователя осуществляется *командным процессором* ОС. Эта часть операционной системы выполняет важную функцию поддержки взаимодействия с пользователями. Всякая ОС имеет командный язык, посред-

ством которого пользователь может инициировать те или иные действия – обращение к директорию, разметку внешних носителей, запуск программ и др. Кроме ввода отдельных команд, которые немедленно выполняются, имеется возможность составления небольших программ на командном языке, с помощью которых можно задать целую последовательность действий. Командный процессор в некоторых ОС позволяет создать удобную операционную обстановку для конкретного пользователя, избавив его от утомительных служебных операций.

СКОЛЬКО ЗАДАЧ МОЖЕТ РЕШАТЬСЯ НА ПЭВМ ОДНОВРЕМЕННО?

Большинство простых ОС обеспечивает такую работу машины, при которой в каждый период времени на ней решается только одна задача. Примером такой задачи может являться процесс редактирования текстов, или работа какого-либо транслятора, или печать текста на принтере. При решении такой задачи оперативная память машины и процессор не могут быть заняты другой работой.

Некоторые типы ОС ориентированы на одновременное обслуживание нескольких задач. При этом имеется в виду возможность запуска одной или нескольких программ с соответствующим распределением оперативной памяти между ними, переключением центрального процессора и других устройств с обслуживания одной задачи на другую; организации обмена сообщениями между ними; синхронизации и др. Типичное использование возможности многозадачного режима – запуск на фоне диалоговой работы пользователя программы печати на принтере или программы поддержки связи с локальной сетью. Довольно часто под многозадачным режимом подразумевается одновременное обслуживание нескольких пользователей, сидящих за отдельными терминалами. Однако для ПЭВМ среднего класса, имеющих всего один дисплей и клавиатуру, такой режим работы не нужен, и специальная поддержка его средствами ОС отняла бы слишком много ценных ресурсов.

Кроме рассмотренных общих функций, ОС иногда обеспечивают программную поддержку некоторых сервисных функций, таких, как вывод на экран дисплея простейших геометрических фигур, поддержка дисплейных окон и др. Особую функцию несут программы, позволяющие использовать ПЭВМ в качестве терминала большой ЭВМ. Такие программы называют эмуляторами терминалов. Реализуются они в виде специальных драйверов в рамках стандартной ОС. Программа-эмулятор воспринимает от пользователя команды, обращенные к другой машине,

перерабатывает их в соответствующее внутреннее представление и посылает через линию связи в другую машину. Кроме того, эта программа должна принимать сообщения от другой ("главной") машины и выдавать их на дисплей ПЭВМ. Важнейшая функция программ-эмуляторов – обеспечение пересылки файлов между машинами в обоих направлениях.

Таковы типичные функции и состав операционных систем для ПЭВМ. Чтобы дать читателю более детальное представление об ОС, рассмотрим подробнее несколько конкретных систем.

ОПЕРАЦИОННЫЕ СИСТЕМЫ СЕМЕЙСТВА CP/M

CP/M положила начало созданию ОС для микроЭВМ. Она была разработана в 1974 г., после чего была установлена на многих 8-разрядных машинах. В рамках этой ОС было создано очень большое количество системных и прикладных программ, включая трансляторы с языков Бейсик, Паскаль, Си, Фортран, Кобол, Лисп, Ада и многих других, текстовые и табличные процессоры, системы управления базами данных, графические пакеты, символьные отладчики и другие проблемно-ориентированные программы. Успех системы в значительной степени был обусловлен ее предельной простотой и компактностью, возможностью быстрой настройки на разные конфигурации ПЭВМ. Первая версия системы занимала всего 4 Кбайта, что было весьма важно в условиях ограниченных объемов памяти первых персональных компьютеров. Затем система была реконструирована с целью обеспечения большей независимости от состава и параметров внешних устройств, применяемых на ПЭВМ. В современных версиях этой системы настройка на конкретную архитектуру машины обеспечивается с помощью специального текстового файла, описывающего конфигурацию подключенных внешних устройств.

CP/M постоянно хранится на диске и состоит из трех частей: базовой системы ввода/вывода, базовой дисковой операционной системы и командного процессора. Эти части содержатся в специальных файлах с именами BIOS, BDOS и CCP. В начале работы при включении машины указанные файлы загружаются в оперативную память. При этом они располагаются на старших адресах, оставляя младшие адреса для программ пользователя, загружаемых с диска. Требуемая для работы память невелика – в пределах 16 Кбайт.

Базовая система ввода/вывода (БСВВ) содержит в основном драйверы внешних устройств. Это единственная часть CP/M, которая требует настройки на конкретное оборудование. Таблицы, описывающие характеристики используемых дисковых накопителей, параметры дисплея, клавиатуры и последовательного

канала, формируются при генерации ОС, во время которой происходит перекомпиляция БСВВ с использованием специального файла конфигурации CONFIG.CPM. При установке нового устройства достаточно внести в CONFIG.CPM соответствующую информацию, скомпилировать новый вариант БСВВ, и система оказывается готовой к работе.

Часть CP/M, называемая базовой дисковой операционной системой (БДОС) содержит функции управления файловой системой и общего управления машиной (таких функций около 40). Файловая система CP/M довольно ограничена. Имена всех дисковых файлов хранятся в одном большом каталоге, называемом директорием. Директорий можно разделить на несколько пользовательских областей, которые идентифицируются порядковыми номерами.

Такая организация файловой системы не очень удобна, так как не позволяет структурировать хранимую информацию. Весьма ограничены возможности улавливания ошибочных ситуаций; диагностические сообщения также очень лаконичны. Однако компенсацией за все эти ограничения является компактность системы и удобство ее настройки на заданную конфигурацию.

Командный процессор (КП) имеет средства для обработки лишь нескольких встроенных команд. Сюда относятся команды настройки на рабочий диск, команда USER – настройка на нового пользователя, DIR – выдача директория диска, TYPE – вывод содержимого файла на экран дисплея, REN – переименование файла, ERA – удаление файла. Это самый минимальный набор команд, которыми приходится пользоваться при общении с ОС. Другие команды (а их число в развитых системах, например в UNIX, может составлять несколько сотен) реализуются в виде независимых программ, работающих под управлением CP/M.

Некоторые из этих служебных программ всегда поставляются вместе с системой. Такова, например, программа PIP, обеспечивающая пересылку содержимого файлов между "источником" и "назначением", программа STAT, позволяющая просматривать и изменять атрибуты файлов на дисках, FORMAT – программа форматирования (начальной разметки) дисков и некоторые другие. Любая программа, в том числе реализующая какую-либо команду общего пользования, запускается очень просто: пользователь в ответ на "приглашение" системы вводит имя программы и параметры, если они необходимы, после чего система автоматически находит соответствующую программу на диске, загружает ее в память и начинает исполнение. Возможно объединение нескольких команд в одном файле и последующее исполнение такого "ко-

мандного файла"; при этом последовательно исполняются все записанные в нем команды.

Система CP/M послужила образцом и основой для создания целого семейства ОС как для 8-, так и для 16-разрядных ПЭВМ. Многопользовательская версия этой системы называется MP/M, а для 16-разрядных ПЭВМ соответствующие версии называются CP/M-86 и MP/M-86. Эти версии по всем функциям похожи на CP/M и MP/M, но ориентированы на работу с 16-разрядным микропроцессором 8086. Еще две версии этой системы – Concurrent CPM и Concurrent DOS – поддерживают многозадачный режим работы.

Операционные системы семейства CP/M оказали заметное влияние на техническую политику многих разработчиков ПЭВМ. Некоторые из них выбирали CP/M или CP/M-86 в качестве основных ОС для своих машин, другие создавали собственные системы по образу и подобию CP/M, давая им другие названия. До сих пор для дешевых 8-разрядных персональных компьютеров CP/M считается одной из наиболее подходящих операционных систем.

ОПЕРАЦИОННАЯ СИСТЕМА MS-DOS (ДОС-16)

Для 16-разрядных ПЭВМ, использующих 16-разрядные микропроцессоры типа 8088 и 8086, безусловно предпочтительней ОС типа MS-DOS. Версия этой системы для отечественных ПЭВМ называется ДОС-16. Принятие ДОС-16 в качестве главной ОС для персональных компьютеров, имеющих широкое распространение, является стимулом для многих программистов к созданию для нее многочисленных инструментальных и прикладных систем. В результате эта ОС приобрела статус фактического стандарта операционной системы для 16-разрядных персональных компьютеров. Это, в свою очередь, привлекло к ней интерес со стороны массовых потребителей.

К основным достоинствам ДОС-16 относятся:

возможность организации многоуровневых директориев,

возможность подключения пользователем дополнительных драйверов внешних устройств;

возможность работы со всеми последовательными устройствами как с файлами;

развитый командный язык;

возможность запуска фоновых задач одновременно с диалоговой работой пользователя и др.

Для работы ДОС-16 требуется существенно больше оперативной памяти, чем для CP/M, – около 60 Кбайт. В настоящее время для ДОС-16 разработан огромный фонд программного обеспечения. Есть трансляторы для практически всех популярных языков

высокого уровня, включая Бейсик, Паскаль, Фортран, Си, Модула-2, Лисп, Лого, Апл, Форт, Ада, Кобол, ПЛ-1, Пролог, Смолток и др.; причем для большинства языков существует несколько вариантов трансляторов. Имеются инструментальные средства для разработки программ в машинных кодах – ассемблеры, символьные отладчики и др. Эти инструментальные средства сопровождаются редакторами, компоновщиками и другими сервисными системами, необходимыми для разработки сложных программ.

Кроме системного программного обеспечения, для ДОС-16 создано множество прикладных программ, которые могут работать на разных машинах, использующих эту ОС. Здесь, однако, следует иметь в виду важное обстоятельство, касающееся переносимости программ. Одни прикладные программы могут быть написаны "чисто" с точки зрения ОС, т.е. обращаться только к функциям ОС для выполнения любых действий. Такие программы без всяких затруднений могут быть перенесены с одной марки машины на другую; важно только, чтобы эти машины работали с одной и той же версией ДОС-16.

Имеются, однако, программы, которые с целью повышения эффективности отдельных операций обращаются непосредственно к аппаратным средствам ПЭВМ в обход функций ОС. С такими программами, а их не так уж мало, возникают проблемы при попытке переноса на другую модель ПЭВМ. К сожалению, довольно часто в прикладных пакетах, пользующихся наибольшей популярностью у потребителей, применяются "запрещенные" с точки зрения ОС приемы; это делается для достижения высоких показателей эффективности.

СОСТАВ И ФУНКЦИИ ДОС-16

В состав ДОС-16 входят примерно такие же компоненты, как и в CP/M, но на них возложено гораздо больше функций. Основные части ДОС-16:

базовая система ввода/вывода, находящаяся в постоянном запоминающем устройстве ПЭВМ, содержит драйверы основных физических устройств – гибких и жестких дисков, дисплея, клавиатуры, принтера, последовательных коммуникационных каналов и внутренних часов;

расширение базовой системы ввода/вывода – часть, которая служит дополнением к БСВВ и, кроме того, несет ряд дополнительных функций;

система обработки прерываний обслуживает обращения к ДОС-16 со стороны прикладных программ, распределение ресурсов между задачами и работу файловой системы;

командный процессор обеспечивает интерпретацию команд-

ного языка, загрузку программ в оперативную память и их запуск.

Базовая система ввода/вывода (БСВВ) содержит драйверы стандартных внешних устройств ПЭВМ, а также конфигуратор системы. В отличие от CP/M в ДОС-16 данная подсистема представляет собой не цельную программу, а совокупность нескольких независимых подпрограмм-драйверов. При необходимости включения нового или замены одного из имеющихся драйверов нужно изменить соответствующую подпрограмму, либо воспользоваться возможностью присоединения к системе новых драйверов извне. Именно для этого служит другая часть ДОС-16 – расширение базовой системы ввода/вывода (РБСВВ). Основная часть БСВВ, находящаяся в постоянном запоминающем устройстве ПЭВМ, не может быть изменена программным путем. Зато РБСВВ находится в отдельном файле на диске, который, вообще говоря, можно заменить другим файлом.

ДОС-16 позволяет задавать конфигурацию системы во время ее начальной загрузки. При этом в специальном файле конфигурации CONFIG.SYS можно указать дополнительные драйверы внешних устройств, "свой" командный процессор, количество буферов для ускорения ввода/вывода и другие важные параметры, влияющие на работу. Эти параметры обрабатываются при запуске системы программами БСВВ и РБСВВ. Таким образом, ДОС-16 обеспечивает не только встроенные операции управления внешними устройствами, но и позволяет динамически настраивать систему на требуемые условия работы.

Система обработки прерываний (СПР) является главной частью ДОС-16. В ней имеется множество системных подпрограмм, которые срабатывают при обращениях к ДОС-16 со стороны пользовательских программ. Такие обращения оформляются специальным образом и называются прерываниями ДОС. Подпрограммы, входящие в эту часть ДОС-16, содержат функции управления файловой системой, поддержки обмена с последовательными устройствами, динамического распределения памяти, физического доступа к дискам, реакции на особые ситуации и др.

Командный процессор (КП) обеспечивает поддержку диалога с пользователем, а также загрузку и запуск прикладных программ. Он поддерживает также механизм возврата из программ после их окончания и некоторые типы реакции на ошибочные ситуации, возникающие в пользовательских программах.

КАК ИСПОЛНЯЮТСЯ КОМАНДЫ?

Рассмотрим, каким образом пользователь осуществляет доступ к файлам и запускает программы на исполнение. Каждый файл имеет имя и тип. В одних случаях для точной идентификации файла следует указывать и имя, и тип, разделяемые знаком "точка", например, GRAPHICS.COM; в других случаях достаточно указывать только имя.

В ДОС-16, как и в других типах операционных систем, физические дисковые накопители именуются буквами A: B: C:... Кроме того, в связи с иерархической организацией файловой системы имеет значение имя директория, в котором зарегистрированы те или иные файлы. Прежде чем пользователь приступает к какой-либо работе, он должен настроиться на "текущий" (рабочий) накопитель и "текущий" директорий. Для этого достаточно сначала набрать имя накопителя, например, C:

После этого, если требуется, можно настроиться на текущий директорий, например, командой CD \WORK.

С этого момента любой файл будет отыскиваться или создаваться в текущем директории WORK, если только пользователь не снабдит имя файла специальным префиксом, указывающим на необходимость поиска его на другом накопителе и/или в другом директории.

Когда пользователь в ответ на "приглашение" системы вводит с клавиатуры некоторое имя, то система считает, что это – либо имя встроенной команды, либо имя программы, которую можно запустить на исполнение. Многие из них требуют задания параметров, которые должны быть введены вслед за именем. Например, команда FORMAT A: /S говорит о необходимости запустить программу форматирования (разметки) диска A: и после форматирования перенести на него копию операционной системы. Сама программа FORMAT должна быть взята из текущего директория данного накопителя или из другого директория, который перед этим может быть указан специальной командой PATH.

Встроенные команды ДОС-16 реализуют операции, которые используются чаще всего. Встроенными являются следующие основные команды (даны русские и английские названия команд):

КАТ		DIR		– выдача каталога (директория),
КОП		COPY		– копирование файлов,
ВЫВ		TYPE		– выдача текста файла на экран,
ИМЯ		RENAME	REN	– переименование файлов,
УДАЛ	УБР	ERASE	DEL	– удаление файла,

НОВКАТ	НК	MKDIR	MD	– создание нового каталога,
СМЕНКАТ	СК	CHDIR	CD	– смена текущего каталога,
УБРКАТ	УК	RMDIR	RD	– удаление каталога,
ПРИГЛ		PROMPT		– задание "приглашения" для ввода,
ДАТА		DATE		– выдача/установка даты,
ВРЕМЯ		TIME		– выдача/установка времени,
МАРШ		PATH		– задание маршрута по файловой системе,
ВЫДМЕТ		VOL		– выдача метки диска,
ЭКР		CLS		– гашение экрана дисплея,
ВЕРС		VER		– выдача номера версии ДОС-16,
КОНСОЛЬ		CTTY		– переназначение дисплея и клавиатуры,
ПАРАМ		SET		– задание параметров операционной среды.

Пример обращения к команде копирования файлов:

КОП *.EXE B:

Здесь предполагается, что все файлы текущего директория, имеющие тип EXE, нужно переписать на диск, установленный на устройстве B:. Знак * означает "любое имя". В качестве первого параметра этой команды можно было бы указать конкретное имя или, например, такое сокращенное имя: PR*.* Это означало бы "все файлы, имена которых начинаются с букв PR". Задание вместо конкретного имени комбинации *.* означало бы соответственно "все файлы".

Пример другой встроенной команды:

ВЫВ CONFIG.SYS

Данная команда осуществляет вывод текста файла CONFIG.SYS на экран.

Помимо встроенных команд, стандартный комплект ДОС-16 сопровождается набором служебных программ, "работающих" подобно встроенным командам. Наиболее часто используемые команды этого типа:

ФОРМАТ	FORMAT	– разметка диска,
ДИСК	CHKDSK	– выдача параметров заполнения диска,
РЕЖИМ	MODE	– изменение режимов работы стандартных устройств,
СИСТ	SYS	– копирование ДОС-16 на другой диск.

Три типа файлов – EXE, COM и BAT – являются "исполняемыми". Это значит, что достаточно указать имя такого файла в качестве команды, и система после загрузки его в ОЗУ сделает попытку запустить его на исполнение. Файл типа EXE обычно получается после трансляции программы с языка высокого уровня и работы компоновщика. В исходном виде он не пригоден к немедленному исполнению, так как требует настройки на место фактического размещения в оперативной памяти. Это делает сама ОС, если файл устроен правильно.

Файлы типа COM не требуют подобной настройки, так как в них все адреса имеют фактические значения. Такой файл может исполняться сразу же после загрузки в ОЗУ. Файл типа EXE при определенных условиях легко преобразовать в файл типа COM.

Особый тип исполняемого файла представляют *командные файлы*, имеющие тип BAT. Эти файлы содержат последовательности команд, которые понимает и интерпретирует командный процессор. Команды могут быть встроенными, или же они реализуются как автономные программы. Кроме имен команд и прикладных программ, в командных файлах могут употребляться специальные команды и выражения для организации простейших форм диалога с пользователем. Имена этих команд: ЭХО, ПАУЗА, ЕСЛИ, ДЛЯ, НА, СДВИГ. Кроме того, в командных файлах могут употребляться метки и комментарии. Таким образом, пользователь может составить достаточно сложную управляющую программу, которая будет заменять ручной ввод команд и обеспечивать определенную автоматизацию.

В ДОС-16 принято особое соглашение, – если на диске, с которого происходит загрузка системы, в корневом директории имеется командный файл AUTOEXEC.BAT, то он автоматически начинает исполняться, как только закончатся все начальные операции по запуску системы. В этом файле можно задать различные начальные установки: дату, время, определенный вид "приглашения" для ввода команд, операционную обстановку, для различных прикладных пакетов, стандартный "путь" для поиска файлов общего пользования и др.

Программа в ДОС-16 может запустить любую другую программу. Для запуска новой программы достаточно специальным образом обратиться к ДОС-16, указав имя файла, в котором находится необходимая программа. По окончании запущенной программы управление вернется к исходной программе.

Еще одна замечательная возможность связана с изменением направления ввода/вывода для любой программы. Например, чтобы выдать текущий каталог не на дисплей, как обычно, а на принтер, достаточно дать команду

KAT > PRN

Знак ">" означает, что вся информация, выводимая командой КАТ (а на ее месте может быть любая программа), должна быть направлена на PRN, т.е. на стандартный принтер. Вместо PRN можно указать имя любого файла или имя любого последовательного устройства.

Можно заставить целые цепочки программ передавать друг другу информацию. Так, по команде PROG1 | PROG2 | PROG3 выходная информация, вырабатываемая программой PROG1, поступит на вход PROG2, а ее выход с свою очередь свяжется со входом PROG3.

Указанные возможности, как и многие другие свойства ДОС-16, заимствованы в системе UNIX. Это отражает тенденцию в сближении этих двух известных систем. Таким образом, ДОС-16 является важнейшей составной частью персонального компьютера, поддерживающей, с одной стороны, работу прикладных программ и обеспечивающей, с другой стороны, исполнение управляющих команд.

Однако сама по себе никакая ОС не дает возможности решать какие-либо практически полезные задачи. Для этого необходимо создание прикладных систем. Но чтобы разработать прикладную систему, нужны инструментальные средства. В следующем разделе дается краткое описание таких средств.

КАК СОЗДАЮТСЯ ПРИКЛАДНЫЕ ПРОГРАММЫ?

Разработка системного и прикладного программного обеспечения для персональных компьютеров осуществляется на основе инструментальных средств, к которым относятся:

макроассемблеры (машинно-ориентированные языки);

трансляторы с языков высокого уровня;

средства редактирования, компоновки и загрузки программ.

Указанные программные средства нужны программистам для того, чтобы составлять другие программы – именно поэтому они и называются инструментальными. Это как бы "средства производства" в информатике, в то время как собственно прикладные системы можно считать "средствами потребления", если пользоваться терминологией политэкономии.

С каждой операционной системой связывается относительно стандартный набор инструментальных средств. Различия имеются лишь в версиях языков высокого уровня и в форматах объектных и загрузочных модулей, получающихся после трансляции и компоновки программ.

БЕЙСИК – ЯЗЫК МАССОВОГО ПРОГРАММИРОВАНИЯ НА ПЭВМ

Основные инструментальные языки высокого уровня, используемые на персональных компьютерах – Бейсик, Паскаль, Си, Фортран. Исторически одним из первых языков высокого уровня для микро-ЭВМ стал Бейсик. Этот язык прост в освоении; его интерпретатор обычно занимает небольшой объем памяти – в пределах 32 Кбайт. В то же время на Бейсике можно писать довольно сложные программы с использованием всех возможностей, предоставляемых аппаратными средствами. В язык, как правило, встраиваются удобные функции для работы с экраном дисплея, клавиатурой, внешними накопителями, принтером, коммуникационными каналами. В связи со своей особой ролью Бейсик считается необходимой принадлежностью персонального компьютера и, как правило, записывается в ПЗУ машины, становясь как бы частью аппаратуры. В настоящее время имеется довольно много реализаций этого языка, но наиболее популярна версия Бейсика для ДОС-16.

Чаще всего начинающие программисты пользуются языком Бейсик для составления своих первых программ, но, переходя впоследствии к более профессиональному программированию, нередко продолжают оставаться приверженцами Бейсика. Одна из причин популярности этого языка лежит в способе составления и отладки программ. Дело в том, что транслятор Бейсика работает обычно в режиме интерпретации. Это означает, что пользователь может ввести с клавиатуры текст оператора или выражения и немедленно его выполнить. Но можно и не выполнять оператор немедленно, а составить сначала всю программу или ее часть, а уж затем выполнить ее. При ошибках введенный текст исправляется с помощью встроенного редактора и вновь выполняется. Все это способствует сокращению характерного цикла в работе программиста: составление программы – пробное исполнение – исправление ошибок – повторное исполнение. Этот режим очень удобен при разработке небольших программ. Однако, интерпретирующий режим имеет неизбежный недостаток – программы работают существенно медленнее, чем в случае использования трансляторов компилирующего типа, как, например, для языков Паскаль и Си. В последние годы основные версии языка Бейсик стали снабжаться трансляторами обоих типов – интерпретаторами и компиляторами. При этом на этапе составления и отладки программы используются преимущества интерпретационного режима, а после завершения отладки всех частей программа компилируется, порождая эффективные машинные коды. Появление компиляторов поставило Бейсик в ряд с другими языками высокого уровня и придало ему дополнительную популярность.

В школьном образовании Бейсику отводится ведущая роль как языку обучения основам профессионального программирования. Именно поэтому для школьных машин предусмотрена обязательная реализация Бейсика.

ПАСКАЛЬ И СИ – ЯЗЫКИ ПРОГРАММИРОВАНИЯ ДЛЯ ПРОФЕССИОНАЛОВ

Языки Паскаль и Си чаще всего используются профессиональными системными программистами для разработки системных и прикладных программ. Оба эти языка позволяют работать с данными сложной структуры, обеспечивают контроль типов данных, имеют развитые средства для выделения отдельных частей программ в процедуры. Трансляторы с этих языков работают в режиме компиляции, что позволяет создавать эффективные машинные программы. Важным средством для построения больших программных систем является модульность, т.е. возможность независимой разработки отдельных частей программ и последующего их связывания в единую систему. Все эти особенности способствуют тому, что именно на Паскале и Си разрабатывается большинство крупных программных систем для персональных компьютеров.

Следует отметить, что между указанными языками, несмотря на общее сходство, имеются существенные различия. Паскаль является более классическим языком программирования, который первоначально (в 1969 г.) был создан как учебный язык и лишь спустя 5–6 лет приобрел популярность как отличный инструмент для решения серьезных задач. Программы на Паскале понятны любому программисту-профессионалу, в то же время они транслируются в эффективные машинные коды. В настоящее время разработано более десятка разных трансляторов с Паскаля для ПЭВМ.

Язык Си в отличие от Паскаля с момента своего появления (1972 г.) был ориентирован на разработку системных программ. Он послужил главным инструментом для создания операционной системы UNIX. В этом языке имеются более гибкие средства для эффективного использования особенностей аппаратуры, чем в Паскале. Благодаря этому порождаемые машинные программы, как правило, более компактны и работают быстрее, чем Паскаль-программы. С другой стороны, синтаксис языка Си менее прозрачен чем Паскаля; возможностей для внесения ошибок больше; тексты программ читаются с трудом. В связи с этим язык Си применяется главным образом для создания системных и прикладных программ, в которых скорость работы и объем программ являются критическими параметрами.

МОДУЛА-2, ФОРТРАН И ДРУГИЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Стремление к созданию подлинно универсального и эффективного инструмента системного программирования привело к разработке нового языка – Модула-2. Он разработан известным ученым Н. Виртом – автором Паскаля. Наследуя лучшие черты Паскаля, в том числе его ясный синтаксис, Модула-2 обладает лучшими средствами для разработки больших программных комплексов и позволяет более эффективно использовать особенности аппаратуры. Таким образом, этот новый язык призван заполнить брешь между Паскалем и Си. По мере появления хороших трансляторов и приобретения опыта работы с ними Модула-2 может занять ведущее место в иерархии языков высокого уровня для ПЭВМ.

Язык Фортран как один из самых старых языков высокого уровня активно используется и на персональных компьютерах. Главным образом он применяется в прикладных системах, ориентированных на научные исследования, автоматизацию проектирования и другие области, где уже накоплены обширные стандартные библиотеки программ. Имеется несколько версий этого языка; наиболее популярен Фортран-77.

Другой "старый" язык – Кобол, широко распространенный на больших и средних машинах, на персональных компьютерах почти не используется, хотя для него разработано несколько трансляторов. Дело в том, что вместо Кобола в задачах экономического и управленческого характера с гораздо большим успехом могут использоваться интегрированные системы, базы данных и другие типы прикладных систем.

Кроме основных названных языков высокого уровня, следует упомянуть другие языки, получившие распространение на ПЭВМ: Лого, Лисп, АПЛ, Форт, Пролог. Язык Лого создан с целью обучения детей младшего возраста основам алгоритмического мышления и программирования. Этот язык реализован для большинства персональных компьютеров, применяемых в школах.

Язык Лисп является идеальным инструментальным средством для исследований в области искусственного интеллекта. Имеется несколько реализаций Лисп-трансляторов для персональных компьютеров разных классов. Другой язык, применяемый при разработке систем искусственного интеллекта, – Пролог. Этот сравнительно молодой язык, в основе которого лежит аппарат математической логики, позволяет разрабатывать на основе ПЭВМ экспертные системы, базы знаний и прикладные пакеты.

Язык АПЛ наиболее пригоден для инженерных расчетов в задачах, связанных с операциями над векторами и матрицами. Однако для удобной работы с этим языком требуется довольно большой набор специальных символов, которые приходится наносить на

клавиши стандартной клавиатуры. Это приемлемо лишь при систематическом использовании данного языка.

Язык Форт обладает такой структурой, что для него может быть создан очень компактный транслятор (объем требуемой памяти для транслятора и редактора менее 15 Кбайт). Это стимулировало появление Форта еще на 8-разрядных персональных компьютерах первого поколения. Однако программирование на этом языке требует специальных навыков, поскольку базируется на особой нотации – "обратной польской записи". В связи с этим, несмотря на ряд преимуществ, Форт получил распространение лишь в кругу своих активных сторонников.

В завершение краткого обзора языков высокого уровня необходимо упомянуть язык Ада. Этот язык был создан для разработки больших программных систем, в том числе для решения задач реального времени. Однако, несмотря на попытки придать ему статус универсального языка для всех применений, значительного распространения на ПЭВМ язык Ада не нашел; главная причина этого лежит, по-видимому, в его чрезмерной сложности.

КАКОЙ ЯЗЫК ЛУЧШЕ?

Когда у пользователя возникает необходимость составления программы для решения какой-либо задачи, встает вопрос о выборе для этой цели языка программирования. Во многих случаях решающими оказываются очень простые "земные" факторы – доступность трансляторов и умение человека составлять программы на том или ином языке. Если, однако, в распоряжении пользователя имеется достаточно большой выбор языков программирования, то следует принимать во внимание такие факторы, как назначение разрабатываемой программы, требуемая скорость ее работы, ожидаемый размер, необходимость сопряжения с программами на других языках программирования, основные типы данных, степень использования аппаратных средств и др.

Возможности языков с точки зрения этих критериев могут весьма сильно различаться, поэтому правильный выбор инструментального языка программирования является, вообще говоря, непростой задачей. Часто эта задача решается, исходя из субъективных оценок и локальной обстановки, нежели на основе реальных количественных и качественных показателей конкретных систем программирования.

ЧТО МОЖНО ДЕЛАТЬ НА ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ?

Для многих людей первое знакомство с персональным компьютером начинается с игры. Действительно, компьютерных игр существует великое множество, и многие из них очень увлекательны и полезны. Однако персональные компьютеры создаются, конечно, не только для игр. На них можно программировать, т.е. составлять и отлаживать программы и, наконец, можно обращаться к готовым программам для решения своих задач. Эти три вида занятий — основные; больше всего времени при использовании персональных компьютеров тратится именно на них.

Прикладные системы образуют уровень программного обеспечения, обращенный к человеку, который обычно не составляет программ, а лишь использует их с целью решения своих задач. В отличие от программистов таких пользователей называют "конечными", подразумевая, что именно они и являются окончательными потребителями тех знаний, которые сосредоточены в памяти компьютера или могут генерироваться во время работы прикладных программ. При общении с прикладной системой пользователю иногда приходится выполнять некоторые простые операции — вводить числа и тексты, просматривать данные, выводить графики, рисунки или чертежи на экран дисплея и на внешние устройства и др. Прикладные системы конструируются таким образом, чтобы создать человеку максимальный комфорт при выполнении таких действий и при этом не требовать от него чрезмерно больших навыков и специальных знаний, не относящихся непосредственно к его профессиональным интересам. Сложилось несколько основных классов прикладных систем, используемых на персональных компьютерах:

прикладные пакеты и программы общего назначения,
проблемно-ориентированные пакеты и программы,
интегрированные прикладные системы.

К пакетам и программам общего назначения, особенно широко применяемым в сфере управленческой и организационной деятельности, относятся:

текстовые процессоры,
программы обработки электронных таблиц,
пакеты графического представления данных (деловая графика),
системы управления данными (базы данных),
системы поддержки коммуникаций.

Текстовые процессоры предназначаются для подготовки всех видов текстовой документации — статей, писем, технических описаний и др. Практически любой документ, который обычно готовится на пишущей машинке, может быть создан с помощью текстового процессора; при этом появляется возмож-

ность многократно исправлять отдельные фрагменты, не вводя заново весь текст, изменять шрифты, вносить рисунки, также приготовленные на ПЭВМ, и, наконец, печатать на принтере в нужном количестве экземпляров. Можно автоматически составлять оглавления документов, проверять правильность написания слов и пр. Таким образом, подготовка текстовых материалов на ПЭВМ может выполняться не только быстрее и эффективнее, чем на пишущей машинке, но и предоставляет новые, недоступные ранее возможности.

Программы обработки электронных таблиц вошли в ряд основных прикладных систем для ПЭВМ с самого начала их массового появления. Электронной таблицей называют способ представления данных, который весьма похож на обычный, ручной, способ представления табличной информации. В памяти машины отображается большая прямоугольная таблица, а на экран дисплея выводится ее часть. При перемещении дисплейного окна вдоль таблицы пользователь может увидеть любую группу ее ячеек. При этом он может вводить в них новую информацию, просматривать содержимое, устанавливать зависимость содержимого одних ячеек от других и др. На основе таких таблиц могут строиться весьма сложные модели, отображающие хозяйственную деятельность предприятий, бухгалтерский учет и др.

Представление данных в виде графиков разного типа является очень наглядным и удобным для их визуального анализа. Поскольку персональные компьютеры, как правило, дают хорошие возможности для работы с графической информацией, системы поддержки "деловой" графики получили широкое распространение и, в частности, используются совместно с системами обработки электронных таблиц и другими системами обработки данных.

Базы данных предназначены для хранения структурированных данных и доступа к ним по запросам от пользователя. Существует несколько десятков систем управления данными на ПЭВМ, различающихся основными параметрами – числом записей в базе данных, числом полей в записи, способом оформления доступа для поиска требуемой информации и др. Ни одна серьезная экономическая модель не может быть построена без хорошей базы данных; необходимы они также для реализации систем автоматизации проектирования и автоматизации научных исследований.

Системы поддержки коммуникаций необходимы для подключения к ПЭВМ различных типов дополнительных внешних устройств, для организации связи между машинами, для поддержки работы ПЭВМ в локальной сети. Эти программы позволяют устанавливать режимы работы последовательных каналов, а также более сложные функции, такие, как поиск нужного абонен-

та в телефонном справочнике, автоматический набор телефонного номера, автоматический ответ на вызов и др. Отдельную группу образуют программы, которые должны поддерживать связь ПЭВМ с научными приборами и установками.

Проблемно-ориентированные пакеты и программы в отличие от программ общего назначения предназначаются для специалистов определенного профиля или узкого класса применений. Число таких программ для персональных компьютеров в настоящее время составляет несколько тысяч. В отличие от программ общего назначения они используют особые методы представления и обработки данных, учитывающие специфику конкретных задач.

Интегрированные системы образуют особую категорию программного обеспечения. Типичная интегрированная система включает текстовый процессор, базу данных, средства работы с таблицами, пакет деловой графики и средства поддержки коммуникаций. Главное внимание в этих системах уделяется тому, чтобы пользователь применял примерно одни и те же приемы работы и мог быстро переключаться с одной группы операций на другую. Еще одно требование – простота действий пользователя при решении простых, часто встречающихся задач и обращение к сложным вариантам работы лишь в редких случаях.

Один из перспективных подходов состоит в том, чтобы предоставить пользователям не готовые интегрированные системы, а удобные средства для их создания. Такие средства трактуются как надстройки над операционными системами, позволяя соединять несколько прикладных пакетов в рамках единой, удобной для пользователей, операционной обстановки.

КАК НАЧАТЬ РАБОТУ НА ПЭВМ?

Трудно ли научиться работать на персональном компьютере? Попробуем описать самые начальные действия человека, который впервые сел за машину и хочет запустить на ней какую-нибудь программу, например популярную игру XONIX. Разумеется, первое, что нужно сделать, это включить машину и дисплей. Машина начинает работать, и на экране дисплея появляются сменяющие друг друга текстовые сообщения. Сначала машина осуществляет самотестирование, и при этом на экран могут выдаваться различные контрольные цифры. Чаще всего тестированию подвергается установленная в машине оперативная память, клавиатура, дисплей. Если что-либо окажется неисправным, то на экран будут выданы соответствующие диагностические сообщения и работу придется прекратить.

Если тестирование проходит нормально, то вслед за этим начина-

ется следующий этап – и н и ц и а л и з а ц и я ОС, – который, как правило, тоже сопровождается выдачей различных сообщений. На большинство из них можно не обращать особого внимания, но некоторые сообщения содержат вопросы, обращенные к пользователю, на которые нужно отвечать. Ответ обычно заключается в том, что пользователь должен набрать на клавиатуре одно или несколько чисел в определенном формате и, нажимая специальную клавишу "Исполнение", заставить машину принять набранное значение (клавиша "Исполнение" располагается в правой части клавиатуры; на ней обычно изображен специальный значок в виде "кочерги").

Одним из вопросов система может запрашивать у пользователя текущую д а т у, следующим вопросом обычно запрашивается текущее в р е м я. Отвечая на эти вопросы, необходимо строго придерживаться того формата, в котором обычно указываются эти величины. Соответствующая подпрограмма обычно сама указывает необходимый формат, используя для этого условные обозначения. Как только пользователь наберет соответствующие цифры (например, 10:35) и нажмет клавишу "Исполнение", в системе зарегистрируется это время, и далее будут работать внутренние часы, пока не произойдет выключение машины или не будет установлено новое время.

В машине могут быть установлены внутренние электронные часы и календарь, питающиеся от постоянной аккумуляторной батарейки. В этом случае дата и время не запрашиваются, поскольку они уже "известны" системе.

Еще один вопрос к пользователю в ходе инициализации может быть связан с о б ъ е м о м в и р т у а л ь н о г о д и с к а. Если в машине установлена достаточно большая оперативная память, (например, более 256 Кбайт), то часть ее можно занять под виртуальный диск. Виртуальным он называется потому, что никакого диска реально не существует, а имеется лишь область в оперативной памяти, которая может использоваться точно так же, как и обычный накопитель на магнитном диске. Изначально виртуальный диск не содержит никакой информации, а после выключения машины все, что туда было записано, пропадает. Тем не менее работать с виртуальным диском в некоторых случаях очень удобно, потому что, во-первых, доступ к нему для записи и чтения файлов производится на порядок быстрее, чем к реальному гибкому или жесткому диску; во-вторых, наличие в системе дополнительного диска всегда создает определенные удобства для работы.

Итак, операционная система при инициализации может запросить у пользователя размер выделяемого виртуального диска в Кбайтах. Пользователь, зная объем физической оперативной памяти и ожидаемые требования к памяти со стороны тех программ, кото-

рые будут далее запускаться, может указать объем виртуального диска.

Наконец, на стадии инициализации система может запросить идентификатор или код пользователя. Идентификатор пользователя бывает необходим для единственной цели – как можно более "гладко" ввести пользователя в привычную операционную обстановку. Дело в том, что операционная система может быть сконфигурирована таким образом, чтобы при начальном входе устанавливались различные параметры операционной обстановки. Сюда относится имя рабочего директория, имена служебных директориев с различными вспомогательными файлами, режим работы дисплея, "почта", пришедшая на имя данного пользователя и др. Можно в порядке начальной установки сразу указать имя первой исполняемой программы, с тем чтобы система автоматически "вошла" в эту программу, не требуя от пользователя каких-либо предварительных команд.

Итак, в процессе инициализации пользователь может ответить на несколько вопросов системы или для простоты дать на них "пустые" ответы. При пустом ответе, который образуется в результате простого нажатия клавиши "Исполнение", система сама по умолчанию установит определенное значение запрашиваемой величины. Если при инициализации не происходит входа в какую-либо прикладную программу, то этот процесс заканчивается выдачей пользователю "приглашения" ко вводу команд. На все описанные выше операции уходит от нескольких десятков секунд до 1–2 мин. Теперь возникает вопрос, что делать дальше?

Стандартное приглашение ко вводу команд имеет в простейшем случае следующий вид:

C>

Такое приглашение говорит о том, что система настроена на рабочий внешний накопитель с именем C:. Напомним, что имена накопителей A: B: C: ... ставятся в соответствие физическим устройствам, причем A: и B: соответствуют накопителям на гибких дисках, C: – накопителю на жестком диске "Винчестер", а D: – виртуальному диску, если он был заказан при инициализации.

Теперь, когда машина готова к работе, все зависит от того, где находится программа, которую хочет запустить пользователь. Допустим, что он пришел со своей дискетой, на которой находится нужная программа. Дискету можно вставить в накопитель A:. Для этого конверт с дискетой вставляется в щель накопителя, опускается шторка, и накопитель готов к работе; – эта операция не сложнее, чем установка кассеты в магнитофон. После этого пользователь должен указать операционной системе на необходимость перехода на рабочий накопитель A:. Сделать это очень просто, – нужно набрать на клавиатуре букву A и двоеточие, а затем нажать

клавишу ''Исполнение'' (вообще, почти любая команда должна завершаться нажатием клавиши ''Исполнение''). На экране эти действия отобразятся в следующую картинку:

C> A:
A>

Теперь приглашение системы указывает, что рабочим накопителем является устройство A:. Все готово к запуску программы с дискеты, установленной на этом устройстве. Однако, прежде чем это сделать, почти любой опытный программист сначала захочет убедиться, действительно ли на этом диске есть нужная программа. Для этого можно воспользоваться встроенной командой КАТ (DIR):

A> КАТ

В ответ на эту команду система начнет читать каталог (директорий) диска A:. Это можно увидеть по горящей сигнальной лампочке, встроенной в накопитель. На экран выводится табличка, строки которой соответствуют файлам, записанным на диск. О каждом файле сообщается следующее: имя, тип, длина в Кбайтах, дата и время создания. Внимательно просмотрев выданный на экран каталог и увидев в нем имя нужной программы (в нашем примере говорится об игровой программе XONIX), можно запускать ее. Для этого просто набирается имя соответствующего файла:

A> XONIX

Программа читается с диска и начинает работать. Цель достигнута!

Рассмотрим теперь несколько иной вариант запуска. Допустим, что нужная пользователю программа находится на жестком диске, т.е. на устройстве с именем C:. Тогда пользователю не нужно переключаться с C: на A:. Можно сразу попробовать запустить программу или дать перед этим команду КАТ, чтобы убедиться в ее присутствии на диске C:. Однако здесь возможны осложнения. Жесткие диски типа ''Винчестер'', как правило, содержат сложную иерархическую структуру файловой системы (рис. 5). Вполне возможно, что нужный пользователю файл находится не в корневом каталоге, где он доступен сразу же, а в одном из подчиненных каталогов. Найти его, вообще говоря, можно путем перебора разных директориев, но лучше всего, конечно, спросить у кого-то из опытных людей, в каком именно каталоге он находится. Допустим, что имя этого каталога ИГРЫ и находится он на первом уровне, т.е. непосредственно под корневым каталогом. Проверить подчиненность одного каталога другому очень просто. Если в каталоге верхнего уровня дать команду КАТ, то все каталоги следующего уровня будут ''показаны'' в нем как простые файлы.

Чтобы после этого ''спуститься'' из каталога верхнего уровня

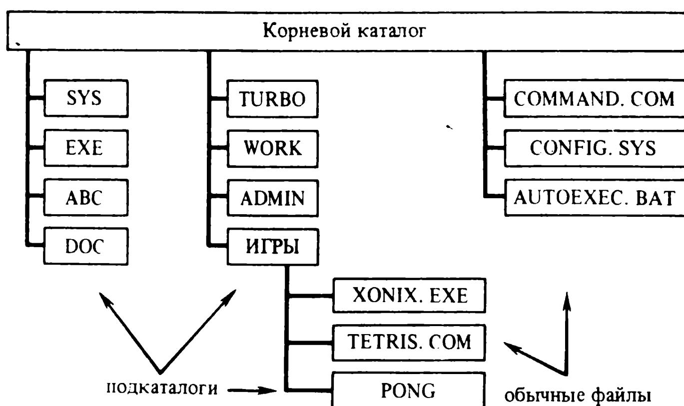


Рис. 5. Типичная структура файловой системы на жестком диске

на один из каталогов следующего уровня, нужно дать команду смены каталога, которая имеет краткое имя СК (CD). В рассматриваемом примере команда смены директория, следующая за системным приглашением, будет выглядеть так:

С> СК ИГРЫ

В новом каталоге ИГРЫ можно опять для контроля дать команду КАТ или сразу же вызвать нужную программу XONIX.

При начальной инициализации иногда устанавливается специальный способ выдачи системного приглашения, так что в нем отображается не только имя рабочего накопителя, но и имя текущего каталога. Такое приглашение имеет для корневого каталога следующий вид:

C:\>

Если же произойдет смена каталога, как в рассмотренном выше примере, то "приглашение" приобретает вид:

C:\ИГРЫ >

Если бы мы погрузились из каталога ИГРЫ в каталог следующего уровня (применяя еще раз команду СК), то соответственно изменилось бы и системное приглашение, став, например, таким:

C:\ИГРЫ\PONG>

Возврат на один уровень вверх осуществляется командой СК .. , а возврат на самый верхний уровень – командой СК \

Таковы минимальные сведения о приемах, которыми необходимо владеть для начального запуска системы, "путешествия" по иерархической файловой системе, поиска и запуска программ.

ПРИГЛАШЕНИЕ К ПРОГРАММИРОВАНИЮ

Г.В. СЕНИН

1. ВОКРУГ И ОКОЛО ПРОГРАММ

Разговор первый

О жесткой и мягкой оснастке

Взглянем на персональный компьютер глазами непосвященного. Мы не найдем в нем ничего необычного: телевизор (или похожий на телевизор монитор), клавиатура, напоминающая клавиатуру пишущей машинки. Человек, работающий с компьютером, нажимает на клавиши, смотрит на экран, т.е. делает вполне обычные вещи. Где же чудеса и загадки, которыми так часто окружают вычислительные машины?

Может быть, у компьютера есть необычные свойства, незаметные при взгляде со стороны? Если мы спросим об этом человека, сидящего за клавиатурой компьютера, то услышим примерно следующее.

– Известные вам приборы умеют делать что-то одно. Телевизор принимает из атмосферы сигналы и преобразует их в изображение и звук. Пишущая машинка печатает знаки, передавая нажатия на клавиши металлическим стержням, ударяющим по бумаге. Магнитофон порождает звучание по записи на магнитной ленте. Все это сложные, но все-таки специализированные действия. Компьютер же может делать разное, сейчас одно, через минуту другое. Происходит это благодаря наличию программы.

– Но и у телевизора есть разные программы, можно переключать...

– Э, нет! Тут совсем другие программы!.. По телевизору вы смотрите то, что вам предлагают, а я могу создать, а потом исполнить на компьютере свою собственную программу.

– Создать? А как она, собственно, выглядит?

– Видите, у меня в руках маленький диск в конвертике? На нем программа. Вставляем диск в гнездо компьютера, программа считывается и исполняется.

– Но ведь то же самое можно сказать и про кассету с видеозаписью?

– Видеомагнитофон только воспроизводит звук и изображение, когда-то записанные, не более.

– А программа что делает – считает, вычисляет?

– Программа может и многое другое. Например, что-нибудь нарисовать на экране. Или, между делом, задать вам вопрос.

– И как ей ответить?

– Для этого служит клавиатура. Нажимаете клавишу – "А", например, – программа ее воспринимает.

– Пишущая машинка тоже воспринимает нажатые клавиши.

– Но она вам ничего не сообщает и никак вашу информацию не обрабатывает. Она только копирует. Кроме того, программу вы легко можете изменить, чуть-чуть подправить, чего не скажешь ни про видеозапись, ни про машинописный текст. Программисты только тем и занимаются, что пишут очередную программу, а потом очень долго ее меняют. Они называют это "редактированием".

Программа не так осязаема, как приборная часть компьютера, ее нельзя пощупать руками. Это информация, которую мы сообщаем компьютеру, и он ее понимает. Английские термины для аппаратуры и программ – hardware, software (буквально: жесткая оснастка, мягкая оснастка) – хорошо отражают суть дела. Программу легко порождать и изменять. На данном компьютере, то есть на одной и той же аппаратуре, можно исполнять тысячи разных программ, и поведение их гораздо сложнее и интереснее, чем воспроизведение звукозаписи.

Разговор второй

О повседневном программировании

– С чем же все-таки можно сравнить программу, раз она не похожа на магнитофонную запись?

– С логической точки зрения программу удобно представлять себе состоящей из отдельных шагов, действий. Запрограммировать – значит предписать действия. Собранные вместе в некоторой последовательности эти действия служат для решения определенной задачи.

– А исполнителем является компьютер? Может быть я чего-то не улавливаю, но все это выглядит простым. Ведь то, о чем вы говорите, встречается в повседневной жизни. Идя в магазин за хлебом, мы тоже выполняем действия, шаги...

– И не только буквальные, физические шаги, но шаги логические...

– Да, да, конечно. Надо взять деньги, а в булочной взамен денег получить буханку хлеба. То же самое, когда я еду на работу: изо дня в день я повторяю одни и те же действия. Прекрасно! Но разве при этом я исполняю программу?

– Почему бы и нет? Эта программа "записана" в вашей памяти...

– Но меня никто не программировал!

– Тогда я попробую привести более убедительную аналогию. Вы, наверное, составляете план ежедневных дел, заносите его в записную книжку, не так ли? Вы не просто оперативно держите программу в памяти, а записываете ее на бумагу. Программист делает в общем то же самое.

– Так-так. Выходит, я программист, я же исполнитель: сам себе предписываю, сам исполняю.

– Эти функции не обязательно совмещены. Вы можете предписывать своему ребенку, вам может предписывать ваш начальник.

– Мы что же друг друга программируем?

– Если хотите, да. Но не будем слишком серьезны, это всего лишь сравнения.

– Давайте же посмотрим, сколь далеко простираются ваши сравнения. Мои ежедневные планы не так уж жестко определены, многие действия зависят от различных обстоятельств: скажем, ближайшая булочная закрыта, придется сесть на автобус, ехать в другую.

– В программах то же самое. Там своя "обстановка", и можно ее учитывать, проверять разные условия. Одна и та же программа будет работать по-разному в зависимости от того, какую информацию ей задать в начале исполнения и по ходу его.

– Ладно, тогда другой вопрос. Свои планы я могу обдумывать с разной степенью подробности. Расписывать по минутам. Составлять план на год, например: "В сентябре купить персональный компьютер". План на всю жизнь, наконец. А как подробно я должен писать программу?

– Это один из самых важных вопросов, хорошо, что вы спросили. Планируя сначала крупными блоками, вы затем переходите к уточнению подробностей. Иначе говоря, вы составляете вашу программу действий последовательно, постепенно, сверху вниз. От программы к подпрограмме, к отдельным, как мы говорим, процедурам. Если в конце концов все свелось к известным, тривиальным действиям, то программа полностью определена, готова к исполнению.

– Тривиальные действия – это какие?

– В повседневной жизни – "спуститься по лестнице", "оторвать билет в автобусе". Для компьютера – "сложить два числа", "передать букву на экран дисплея".

– Неужели придется планировать до таких деталей?

– Все зависит от того, что умеет исполнитель. "Вскипятить воду" – тривиальное действие для взрослого, а ребенку его надо объяснить, разложить на более мелкие: "достать кастрюлю", "налить воду", "зажечь газ" и так далее. Иногда же довольно сложное действие мы включаем в план целиком, не уточняя, потому что

хорошо знаем, как его выполнить, – например "съездить в Ленинград".

Эти параллели человеческого и машинного вызывали у меня некоторое недоверие. Должны же они где-нибудь кончаться? Мы стали выяснять, чем человеческие программы отличаются от машинных.

Первое отличие, которое бросается в глаза, это различие "миров", в которых действует человек и программа, та самая обстановка, о которой мы уже говорили. Мир человека более открыт и непредсказуем, действия человека более многогранны и неформальны. "Мир" программы определеннее, ее действия однозначнее.

С этим отличием связано другое, более глубокое: самоконтроль и корректировка, присущие человеку. Он может отменить или изменить свою программу при неожиданной или неблагоприятной обстановке и начать действовать по-другому, ранее не предусмотренным образом. В нем совмещаются составитель и исполнитель, благодаря чему многие действия выполняются с ходу, без предварительного плана.

Программа не обладает такой гибкостью. Этап составления программы предшествует этапу исполнения. Если встречаются не предусмотренные программистом обстоятельства, то возникает ситуация "ошибки", и дальнейшее исполнение программы либо невозможно, либо происходит непредвиденным образом. В отличие от человека машина-исполнитель совершенно к этому равнодушна, в ней не шевельнется никаких отрицательных эмоций.

У человека, правда, тоже встречается "машинальное" исполнение. Некоторые действия от частого повторения приобретают автоматизм, становятся рутинными. Если вы в автомат метро вместо пятачка пытаетесь засунуть ключ от квартиры, значит у вас включилась не та "рутина". (В одном из языков программирования, о них речь будет ниже, программа обозначается как раз словом "рутина".) Но человек очень редко попадает в ситуацию, где он совершенно не знает, что предпринять, а с машинными программами это случается сплошь и рядом. И тут требуется помощь со стороны программиста.

Разговор третий

О битах и байтах

– Вы показывали маленький диск, на котором была записана программа, потом говорили, что программист составляет программу на бумаге. Как она попадает на дискету и в каком виде там хранится?

– Внешне это похоже на обычную магнитофонную запись, но

только внешне. (Программу можно, кстати, записать и на обычной магнитофонной ленте.) И то, и другое – информация, однако звукозапись построена по непрерывному, аналоговому принципу: намагниченность ленты соответствует амплитуде и частоте звуковых колебаний. Программа же записана, закодирована в виде цифр. Прослушать ее как обычную звукозапись нельзя. Цифровое кодирование, впрочем, применяют уже и для звуко- и для видеозаписи.

– Всюду говорят и о цифровом телевидении. Тем интереснее узнать об этом побольше. То, что единица информации – бит, мне известно, но я плохо увязываю это с тем, что сам понимаю под информацией.

– А что вы под этим понимаете?

– То, что мы находим в книгах, слышим по радио, видим по телевидению – это ведь средства массовой информации, не так ли? То, что можно выразить человеческим языком в письме, в телефонном разговоре. Надо сюда добавить, конечно, и числовую информацию.

– Тексты, в частности письменная речь, и числа достаточно легко сводятся к битовой форме, которую информация принимает в машине и на дискете. Попробуем разобраться как. Когда мы узнаем "орел или решка", то получаем информацию в 1 бит. Налицо один из двух равновероятных, взаимоисключающих вариантов, причем ничего третьего быть не может.

– На ребро монета не встает.

– Именно. По такому принципу в древности действовали сигнальные костры: дым означал наступление некоего события. Бит информации – это сигнальный дымок электронного мира: "да–нет", "истина–ложь", "наличие–отсутствие". В машинном устройстве бит информации представлен одним из двух возможных физических состояний (намагниченностью или электрическим напряжением – в зависимости от типа устройства). Удобно обозначать бит информации нулем или единицей.

– Значит, если в трамвае я спрашиваю: "Вы выходите на следующей?" – то в ответ получаю информацию в 1 бит?

– Да, но помните, что третьего не дано. Если вам ответят: "Отстаньте, вы уже спрашивали", то сами уж решайте, положительный это ответ или отрицательный.

– А разве обе возможности здесь равновероятны?

– Давайте отвлечемся от этого. Компьютер просто отличает нуль от единицы и не видит в этом ни вероятности, ни тем более случайности. В жизни мы довольно часто встречаемся с битами: галочка на полях книги, цвет светофора (если "желтого не дано"). Новая нумерация автомашин в отличие от старой содержит лишний бит информации (государственная или личная), который указывается определенным порядком букв и цифр.

– А если информация более сложная, чем "да" и "нет"? Все-таки бывает и желтый цвет светофора, и пометки на полях – разные...

– Информацию о каком-либо объекте или ситуации вы можете представлять в виде ответа на вопрос: "Обладает ли объект таким-то признаком?". Например, о мебели: "Есть ли у нее ножки?" Или о женщине: "Она замужем?" Накопив ответы, вы получите комбинацию из "да" и "нет", которая соответствует вашему объекту. Прикиньте, сколько таких комбинаций будет при двух вопросах?

– При свободном сочетании ответов – каждый с каждым – получим: 00, 01, 10, 11 (если 0 означает "нет", а 1 – "да").

– Это как раз числа от 0 до 3 в двоичной системе счисления, в которой используются только две цифры: 0 и 1. (Поэтому бит называют также двоичным разрядом.) Таким образом, двумя двоичными разрядами можно различить или обозначить 4 объекта, события, ситуации.

– А если у нас только 3 возможности, например "красный", "желтый" или "зеленый"?

– Тогда одного бита мало, а два дают некий избыток, – одна из четырех возможностей лишняя. Для теории информации это существенно, но для программирования – нет, и мы на это не будем в дальнейшем обращать внимания. Посмотрим лучше, что будет, если увеличивать число разрядов.

– 3 бита дадут 8 вариантов, от трех "нет" до трех "да": 000, 001, 010, 011, 100, 101, 110, 111. И дальше, очевидно, по степеням двойки: 16, 32, 64, 128, для 8 разрядов – $256 = 2^8$.

– Стоп. Мы "открыли" хорошо известный факт: любое целое число можно записать в двоичной системе счисления с помощью только нулей и единиц. Мы можем использовать этот факт для того чтобы числами обозначать любые объекты, о которых хотим хранить информацию, то есть попросту перечислять их. Можно, например, перенумеровать буквы.

– Интересно, сколько бит для этого потребуется! Посчитаем. Букв у нас 33, если Е и Ё считать за одну, то как раз 5 разрядов, поскольку два в пятой степени – 32.

– А если различать большие и малые?

– Большие или малые? Это ведь еще один бит: взаимоисключающие варианты, третьего не дано. Итого 6 бит.

– В программах, кроме русских, используют латинские буквы. И притом намного чаще.

– Латинских чуть меньше. Но если грубо, то хватает 7 разрядов.

– Есть еще знаки препинания, арифметические знаки... Удобно и цифры от 0 до 9 кодировать, как буквы, – они часто встречаются в текстах.

– Значит, 8 разрядов, 256 разных знаков. Этого уж наверняка хватит. По-моему, даже с избытком.

— Насчет избытка не беспокойтесь. Программисты используют разные вспомогательные и необычные символы. Но в принципе мы нашли как раз то количество двоичных разрядов, которое нужно для представления самых разных символов, включая буквы двух алфавитов, цифры, знаки препинания. 8 бит образуют новую, более крупную единицу информации, называемую "байтом". В памяти машины каждый из 256 символов хранится в виде определенной комбинации из 8 нулей и единиц, или, что то же самое, в виде двоичного числа. Приняты соглашения о том, какой символ каким числом обозначается. Помните, мы считаем разными буквами "а русское" и "а латинское", "А большое" и "а малое".

— Итак, 1 буква, например "а малое латинское", несет информацию в 1 байт. Верно?

— Верно. По крайней мере с точки зрения программирования, и пусть не придираются специалисты по теории информации. Байт — единица измерения информационной емкости (памяти) компьютера. Для больших объемов информации используются такие единицы, как "килобайт" (1024 байта) и "мегабайт" (1024 килобайта, или чуть больше 1 миллиона байт). Вместо обычной тысячи здесь используется ближайшая степень двойки: $2^{10} = 1024$.

— Давайте переведем эти числа на более близкие понятия. Сколько байт способна хранить дискета?

— Ну, скажем, 360 килобайт.

— То есть больше 360 тысяч знаков. 40 000 — это печатный лист; следовательно, на дискете уместится книга в 9 листов — страниц двести.

— Добавлю, что бывают диски другого типа, емкость которых выше в 25 и более раз. Подробнее об этом рассказано в статье об устройстве ПК. Обычно объем дискеты, а уж тем более большого диска, не отводится под один гигантский текст, а разбивается на порции, которые не обязательно связаны друг с другом. Эти порции, или участки диска, называют ф а й л а м и. Каждому файлу присваивается определенное имя, которое служит для обозначения информации, хранящейся в нем. Когда мы заносим программу на дискету, то должны записать ее в некоторый файл. Имя этого файла будет для нас ассоциироваться с нашей программой, и мы будем указывать его при копировании программы с одной дискеты на другую, при работе с текстом программы ("редактировании") и т.д. Когда программа примет окончательный вид, то для ее исполнения мы зададим имя файла, в котором она содержится.

Подведем итоги. Мы поняли, как записывать буквы и числа в двоичной форме. В процессе работы на компьютере об этом можно не задумываться, что, кстати, программисты и делают. Знайте только, что именно двоичная информация хранится на электронном носителе информации. Это мир из миллионов электрических или

электромагнитных элементов, каждый из которых может быть в одном из двух состояний: ”включен” либо ”выключен”.

– Понятно, все держится на нулях и единицах.

– Могу добавить еще, что нечисловая и нетекстовая информация, о которой мы здесь не говорили, – изображения, картинки – в памяти компьютера тоже кодируется битами.

Кодирование информации в двоичной форме связывает мир людей с миром компьютеров, создает основу для общего языка между ними и позволяет использовать вычислительные возможности машин для решения практических задач переработки информации.

Разговор четвертый

Компьютер и калькулятор

– Когда мы говорили о различных устройствах и сравнивали с ними персональный компьютер, то как-то забыли о микрокалькуляторах. Конечно, компьютер работает с клавиатурой и дисплеем, которых нет у калькулятора. Это несходство бросается в глаза. Но есть и что-то общее?

– Существуют микрокалькуляторы, на которых тоже можно составлять программы. Из всей окружающей нас электроники программируемые калькуляторы по своему принципу стоят ближе всего к компьютерам.

– Действия, которые они исполняют, – одни и те же?

– Здесь нужно говорить более точно. Конечным исполнителем в обоих случаях является микропроцессор – очень небольшое устройство, в половину спичечной коробки. Исполняемые им действия называют командами, или инструкциями.

– В микрокалькуляторе содержится такой же микропроцессор?

– Более простой. Процессор компьютера некоторые действия поручает другим физическим устройствам, например устройству печати, которое снабжено своим микропроцессором. Как мы уже говорили, он может взаимодействовать с клавиатурой и дисплеем, которых у калькулятора вообще нет. Но многие команды у них похожи, например арифметические действия.

– Вы объяснили, что растолковывать программу исполнителю нужно в тех операциях, которыми он владеет. Команды процессору – это как раз такие операции?

– Совершенно верно. Выполняются они по человеческим понятиям очень быстро: тысячи, сотни тысяч в секунду.

– Но это ведь такие примитивные действия! Вспоминаю наш старый разговор. Если бы я планировал поездку на работу до таких деталей, как шаги по лестнице, то я бы туда никогда не попал, до исполнения дело бы не дошло.

— Согласен. Но что касается процесса составления программы, то здесь между калькулятором и компьютером есть существенные различия. Калькулятор живет в мире чисел и принуждает к этому программиста: на нем вы видите только кнопки, обозначающие цифры и элементарные функции, на его небольшой световой панели — только числовая информация. Программу вы должны записать в виде чисел, чтобы было "удобно процессору", да и объем программы ограничен. Вы как программист остаетесь один на один с "железом". Когда-то большие ЭВМ, предки персональных компьютеров, чем-то походили на нынешние калькуляторы, хотя и были несравненно больше и мощнее. Программа составлялась каждый раз заново от начала до конца и записывалась на "автокоде", в командах процессора. Такое программирование напоминало работу кустаря-одиночки.

— Ручная работа? Но она бывает очень высокого качества.

— Зато сложное, крупное изделие изготовить очень трудно. Требовалось делать большие программы, но они составлялись из мелких элементов-команд и потому были плохо обозримыми, а в результате — ненадежными, содержали много ошибок. И самих программ нужно было больше — не хватало производительности. Выход, конечно, нашелся.

— Мануфактуры, фабрики?..

— Не смейтесь, именно так все и было. С соответствующими поправками, конечно. Произошло разделение труда, возникла своя "инфраструктура". Многие функции, которые приходилось повторять в разных программах, выделились и сосредоточились в особой программе, названной "операционной системой". Хранение информации на дисках было одной из важнейших таких функций, и операционные системы взяли на себя всю работу с файлами, о которых мы упоминали выше. Для выполнения общих функций программы стали прибегать к помощи операционной системы. С этого началось развитие "мягкой оснастки" компьютеров. Затем информация стала вводиться в машину не в цифровом, а в текстовом виде, более удобном и понятном для программиста. Тут уж невозможно стало обходиться без полноценной клавиатуры, которая не требуется калькулятору.

— Мы уже обсудили, как можно кодировать текстовую информацию.

— Учтите, что нужно не только кодировать, но и переводить, преобразовывать текст программы в команды процессора. Это задача отдельной программы — программы-посредника. Ее называют транслятором.

— В этом случае я задаю действия не процессору, а программе-посреднику. Но она их не исполняет сама, а переводит в команды процессора. Правильно я понял?

– Правильно, хотя тут есть один нюанс: сразу будут исполнены эти команды или когда-то впоследствии. Мы к этому еще вернемся. Если действия исполняются сразу, то посредника можно считать исполнителем. Во всяком случае транслятор – это главный подрядчик, и тот факт, что он прибегает к помощи субподрядчика-процессора, будем считать его личным делом.

– Операционная система и трансляторы есть и на сегодняшних персональных компьютерах?

– Без них никак не обойтись. Эти средства, которые называют системным программным обеспечением, можно сказать, окутали железо и скрыли его от глаз программиста.

– Теперь я могу "забыть" не только о битах, но и о командах, которые исполняет процессор?

– Вполне. Человек, сидящий за персональным компьютером, взаимодействует не столько с аппаратурой, сколько с различными программами-посредниками, которые воспринимают нажатия на клавиши, выдают изображение на дисплей... короче, значительно облегчают общение с машинным миром.

– Да, теперь я вижу, что если калькулятор – это станок, то компьютер – целый завод.

– Пожалуй, даже завод-автомат под полным управлением программиста. Компьютер мощнее не только по "мягкой", но и по "жесткой" оснастке. Чтобы трансляторы и операционная система могли работать на машине, пришлось увеличить ее память, информационную емкость. Скорость выполнения команд тоже намного выше, чем в микрокалькуляторе.

– Но несмотря на эту кухню, все сводится к тому, что процессор переваривает простенькие команды. Значит то, что пишут про машинный разум...

– Никакого противоречия тут нет, хотя в рассказах про "умные машины" журналисты иногда перебарщивают. Вас не удивляет работа современного завода? Какое сложное оборудование, изощренные операции, какие тончайшие изделия производятся! А ведь если докопаться, где-то на самом нижнем уровне крутятся совсем простые "винтики". Вся видимая сложность от того, что много уровней, много связей. В программах точно так же. Их разум складывается из очень большого числа банальных операций.

– Удивительно, как все это умещается в таких маленьких объемах?

– Современная технология производства микропроцессоров и микросхем действительно крайне тонкое и сложное дело. Ведь ее прогресс буквально из слона сделал муху: вместо прежнего мамонта-компьютера – крошечный микропроцессор.

– Мне осталась не совсем понятной одна вещь. Как программа, которая хранится на дискете, попадает на обработку процессору?

— Диск — это так называемая внешняя, или долговременная память компьютера. Записав на дискету информацию, вы можете перенести ее на другую машину, поставить на книжную полку рядом с книгой, даже послать приятелю по почте. Информация не исчезнет, если, конечно, не сгибать дискету и не гладить ее утюгом. Но непосредственно процессором программа исполняется только в оперативной, быстрой памяти, куда ее надо "загрузить" предварительно с диска. Оперативная память является лишь временным хранилищем программы: программа будет "стерта" оттуда при выключении машины либо при загрузке на ее место другой программы. Мы об этом еще поговорим ниже.

Разговор пятый

Нужно ли знать программирование?

— Скажите, нужно, по-вашему, уметь программировать "нормальному" человеку?

— Полезно знать о программировании хотя бы понаслышке. Ну, а если есть склонность, то почему бы не научиться? Чем вы моложе, чем больше у вас амбиций, тем вам это нужней. Школьнику это необходимо не меньше, чем знать, скажем, о пастбищах Австралии, хоть я и не имею ничего против географии.

— Но всем поголовно программировать не придется?

— Программировать, может быть, и не придется, но работать с готовыми программами придется почти наверняка. А это значит взаимодействовать, общаться, поскольку современные программы, как правило, ведут интенсивный диалог с человеком. А понимать действия программы, ее поведение легче, если знать, как она устроена.

— Тогда естественно спросить: трудно программировать?

— В принципе азы программирования доступны каждому, даже если начинать с нуля. В программировании, как в каждой науке, есть своя высшая математика, но и без нее, будучи, в общем, дилетантом, можно писать собственные программы и решать с их помощью практические, полезные задачи. Графоманство программированию не грозит.

— Почему же?

— Потому что это сфера более точная и определенная. И более прагматичная что ли. Англичане говорят, что для проверки пудинга надо его съесть. Программа же проверяется в процессе исполнения. В программировании есть хорошие критерии. Невнятная, плохо спланированная программа не воодушевит даже автора, а неправильная не станет выполняться вовсе.

— Иначе говоря, программа не просто сочиняется, а составляется, разрабатывается...

– ... проектируется. Конечно. Если говорить о любительском программировании, то лучше всего сопоставлять его с радиолюбительством, например. Технарю здесь больше раздолья, чем гуманитарию. Составить программу значит в каком-то смысле создать механизм, потому что она "работает". Но учтите, что оснастка-то мягкая, никаких механических деталей, почти никаких материальных затрат, и в этом программа сродни чертежу.

– Позвольте, это что же: план дома и дом, слитые воедино? Дом, который меняется при одном прикосновении к плану?

– Если вам не верится, приведу дополнительное свидетельство этому. Программу очень легко скопировать. Трудно создать только первый экземпляр программы, остальные 100, 1000 получаются, можно сказать даром. Попробуйте размножить дом в сотне экземпляров... Есть и другой аспект – логический. Программы – это же воплощенные, готовые к работе, алгоритмы. Математик легко свяжет программу с доказательством теоремы, в высшей степени конструктивным. Поэтому математику легче освоить программирование.

– Знаете, мне самому хочется его освоить, ощутить, что это такое.

– Мы к этому вполне готовы. Я хочу познакомить вас с Бейсиком.

– Очень приятно. А кто это?

– Это язык, на котором мы начнем программировать.

– И наша программа будет работать?

– Давайте попробуем.

2. "ДЕЙСТВУЙ, БЕЙСИК!"

Изучение программирования по-настоящему полноценно, только если под рукой находится компьютер, однако я постараюсь по мере возможностей поделиться с вами теми знаниями, которые извлек из наших встреч с программистом.

Наше первое занятие по программированию началось с того, что мой собеседник произвел небольшие манипуляции на клавиатуре, а затем произнес: "Бейсик к нашим услугам. Следите за тем, что я делаю". Он нажал несколько клавиш, и на экране последовательно появились символы:

```
PRINT 2 + 2
```

Потом он нажал еще одну клавишу, и на экране появилось:

```
4
Ok
```

– Ага! Бейсик выполнил действие.

– В Бейсике действия принято называть операторами. Наш

оператор означает "печатать", точнее "вывести на дисплей". Обозначается он, как и все операторы Бейсика, английским словом, которое можно рассматривать просто как обозначение, если не знаете английского.

– Кроме того, вы задали ему, что напечатать.

– Да, я задал а р г у м е н т оператора. Часто говорят, что оператор применяется к аргументу. Аргумент нужно отделить от оператора пробелом. Для этого на клавиатуре нажимается специальная клавиша, такая же, как на обычной пишущей машинке.

– А зачем вы нажали самую последнюю клавишу?

– Это клавиша ввода. Нажав клавишу, я как бы сказал Бейсику: "Оператор завершен, передаю его тебе для исполнения".

– Я убедился, что Бейсик "знает" арифметику: он выполнил и сложение.

– В сущности, он произвел два действия: сложил два числа и сумму вывел на экран. Но более точно будет сказать, что транслятор Бейсика перевел наш оператор в несколько машинных команд.

– И тут же исполнил.

– Будем считать, что исполнил он сам, хотя в конце концов все исполняет аппаратура. Трансляторы, способные немедленно выполнить оператор (помните наш разговор о трансляторах?), называют интерпретаторами. Транслятор Бейсика принадлежит к этому типу. Интерпретаторы удобно использовать для обучения программированию – сразу виден результат работы программы. Мы будем называть Бейсиком не только язык, но и интерпретатор языка.

– А что означают буквы "Ok"?

– Бейсик сообщил, что действие выполнено и он "ждет дальнейших приказаний". Пока эти две буквы не напечатаны, вы не можете к нему обращаться, он занят своей работой.

Я извлек из урока и другие сведения.

Можно изменить у оператора аргумент, например:

```
PRINT 5*5-4
```

Учитывая, что звездочкой в Бейсике обозначается умножение, вы без труда определите, какой результат я увидел на экране.

Можно применить другой оператор:

```
LPRINT 5*5
```

что вызывает печать на принтере (печатающем устройстве).

Столкнулся я и с первой ошибкой. ("Наверняка не последняя", – ободрил меня мой учитель). Это произошло, когда я сам стал набирать текст оператора на клавиатуре, и на экране получилось:

```
PRITN  2*2
```

а реакцией Бейсика было: СИНТАКСИЧЕСКАЯ ОШИБКА (правда, с моей точки зрения, ошибка была орфографическая).

– Неужели он не мог догадаться, что переставлены буквы?

– Вы очеловечиваете транслятор. Это же, в конце концов, машина, а машине проще дать отказ, чем гадать. Если уж вам так хочется, считайте, что это скромный, порой даже тупой исполнитель, строго следующий букве ваших указаний.

Оператор есть отдельное предложение языка программирования, т.е. того языка, на котором исполнителю "объясняются" его действия. В нашем случае языком программирования является Бейсик. Строго говоря, то, какие операторы допустимы в языке, а какие нет, определяется транслятором. Два разных транслятора иногда "понимают" похожие языки, но некоторые операторы в них могут отличаться. Тогда говорят о диалектах языка. Так существует довольно много диалектов Бейсика – из-за наличия разных трансляторов. Поэтому всякий раз, когда речь заходит о Бейсике, правомерен вопрос: "А у вас какой Бейсик?" (иначе говоря: "Какой диалект воспринимает ваш интерпретатор?"). Это касается и других языков программирования. Бейсик, который осваиваем мы, тоже является диалектом. Когда вы впервые подойдете к машине, то имейте в виду, что некоторые нижеприводимые операторы могут быть не понятны вашему интерпретатору Бейсика.

Затем я освоил другой оператор. Таким же образом, как и раньше, я написал:

```
LOCATE 10,40
Ok
```

и ... ничего не произошло.

– Смысл этого оператора таков: задать позицию экрана, в которую вы собираетесь печатать. Вы задали позицию: Бейсик принял это к сведению.

– Что означают цифры 10 и 40?

– Это два аргумента оператора LOCATE. Они задают позицию в 10-й строке и 40-м столбце. Строки считаются сверху вниз, а столбцы – слева направо, как в книге.

– И сколько всего позиций на нашем экране?

– 25 строк и 80 столбцов. Каждая позиция это место для одного символа – буквы или знака, поэтому она называется еще *знакоместом*.

– А как мне увидеть эффект моего действия?

– Задайте подряд два известных вам оператора. В Бейсике это можно сделать, разделив их двоеточием:

```
LOCATE 10,40:PRINT 2*2
Ok
```

В середине экрана появилась цифра 4.

– Теперь скажите: то, что мы выполнили, это программа?

– Да пока еще нет. То, что мы выполнили, называют *составным* оператором.

– Но тут два оператора.

– Они составляют как бы единое целое, поскольку записаны в одной строке.

– Но ведь программа это и есть последовательность действий, в нашем случае операторов.

– Если это и программа, то очень неполноценная. Чтобы исполнить ее еще раз, придется написать все заново.

– Ага, мы не сумели запомнить нашу программу. А как это сделать?

– Запомнить какой-либо оператор очень легко – нужно перед ним поставить целое число, например:

```
10 PRINT 2* 2
```

В данном случае мы запомнили оператор под номером 10.Другой оператор нужно будет запомнить под другим номером. Оператор при этом не исполняется, ответ "Ok" не появляется. Чтобы различать немедленное исполнение операторов и запоминание будем называть немедленно исполняемые операторы *приказами*.

– А что означают номера?

– Главное, что операторы будут выполняться в порядке своих номеров. В остальном номера более или менее произвольны.

– Я хочу, чтобы оператор указания позиции выполнился перед оператором печати, поэтому задаю его под меньшим номером:

```
5 LOCATE 10, 40
```

И хотя мы *записали* его позже, *выполнится* он раньше. Можно представлять программу Бейсика в виде длинного "полотна", этакой простыни, прошитой горизонтальными строчками, пронумерованными подряд, начиная с 1.

```
1 . . . . . . . . . . . . . . . . . . . . . . . . .
2 . . . . . . . . . . . . . . . . . . . . . . . . .
3 . . . . . . . . . . . . . . . . . . . . . . . . .
4 . . . . . . . . . . . . . . . . . . . . . . . . .
5 LOCATE 10, 40
6 . . . . . . . . . . . . . . . . . . . . . . . . .
7 . . . . . . . . . . . . . . . . . . . . . . . . .
8 . . . . . . . . . . . . . . . . . . . . . . . . .
9 . . . . . . . . . . . . . . . . . . . . . . . . .
10 PRINT 2 * 2
          . . . . . . . . . . . . . . . . . . .
```

и так далее.

Вначале все строчки пустые. Когда мы пишем оператор с номером, он заносится на простыню в нужную строчку. Если в этой строчке уже стоял какой-то оператор, он стирается и заменяется

на новый. В данный момент программа состоит из двух операторов. В любой момент вы можете посмотреть на программу с помощью особого оператора – LIST. На экране изобразятся только непустые строчки:

```
LIST
5 LOCATE 10, 40
10 PRINT 2 * 2
Ok
```

Итак, программа написана, но еще не исполнена. Ситуация, если вдуматься, знакомая. Когда вам сообщают по телефону рецепт нового блюда: взять, налить, насыпать, смешать – вы не начинаете готовить с прижатой к уху трубкой Вы не исполняете, вы записываете. Это очень удобно. Нет масла? – не страшно; "программа" остается в силе, хотя и зависит от некоторых условий. Кроме того, вы в силах потом изменить рецепт: дополнить, улучшить. Наконец, по рецепту в дальнейшем можно действовать многократно. Полотно с операторами для интерпретатора Бейсика – то же, что рецепт блюда для домохозяйки.

Как исполнить программу? Этому служит оператор RUN, означающий "Действуй". По приказу RUN интерпретатор просматривает простыню сверху вниз и пустые строчки оставляет без внимания, а операторы исполняет.

Перед тем, как исполнить нашу программу, вставим в нее еще два оператора.

```
2 CLS
20 END
```

Первый означает "очистить экран"; второй "завершить исполнение программы". Посмотрим на программу целиком:

```
LIST
2 CLS
5 LOCATE 10, 40
10 PRINT 2 * 2
20 END
Ok
```

После выполнения приказа RUN экран погаснет и в 40-м столбце 10-й строки появится знакомый результат.

Каков смысл введения двух дополнительных операторов? Гашение экрана необходимо для того, чтобы получить результат работы программы в "чистом" виде (экран, как правило, "засорен" текстом, отражающим работу по составлению программы, но не относящимся к ее исполнению). Кроме того, это элемент "культуры" программирования: программа одинаково гладко сообщает результат независимо от исходного состояния экрана.

Так оратор откашливается перед выступлением, заботясь о том, чтобы все слова донести до слушателей. Таким же культурным элементом является оператор окончания программы. Правда, в данном случае он носит чисто декоративный характер, как слово КОНЕЦ на последней странице книги. Наша программа завершится даже в его отсутствие – просто исчерпав все операторы. Но во многих других случаях это не так. Что бы вы сказали, обнаружив КОНЕЦ не в конце книги, а в середине? В книгах так не бывает? А в программах бывает, мы это увидим.

Раз уж мы заговорили о культуре программирования, то остановимся еще на одном операторе. Он выглядит чрезвычайно просто:

REM – здесь может стоять любой текст!

– Совершенно произвольный текст?

– Абсолютно произвольный.

– Но как его выполняет Бейсик?

– Очень просто, он его вообще не выполняет! Увидит REM, игнорирует всю строчку и переходит к следующему оператору.

– Но это же бессмыслица! Зачем он тогда вообще нужен?

– Его пишут не для Бейсика, а для человека.

– Но человек же не исполняет программу...

– Зато он ее читает. Оператор REM служит для записи примечаний, комментариев. Вы ведь уже, надеюсь, понимаете, что программа не составляется за один присест. Она пишется, потом меняется, улучшается, пробуется. Программист обнаруживает в ней ошибки, программа исправляется, о т л а ж и в а е т с я. Первоначально писать ее может один человек, пользоваться и обнаруживать ошибки – другой, исправлять и отлаживать – третий. Очень часто отсутствует ярко выраженный момент окончания разработки программы. Короче говоря, программа живет. Эта жизнь немного похожа на жизнь механизма. Хотя программа и не снашивается (в ней нет механических деталей), но постоянно модернизируется. Однако гораздо больше ее жизнь похожа на жизнь книги, рукописи, которая возникает из черновиков, проходит первую, вторую редакцию. Из нее изымаются одни куски, добавляются другие.

Программу читают, как книгу, компонуют, изготовляют, как механизм. Программа таит в себе и авторскую индивидуальность, присущую рукописи, и социальный облик всякого промышленного продукта, созданного коллективным трудом.

Впрочем, я хотел сказать только о пользе оператора REM. Для того, кто первый раз видит программу и хочет в ней разобраться, комментарии и примечания на обычном человеческом языке, поясняющие логику, структуру, конкретные операторы программы, играют неоценимую роль.

– Ваши слова кажутся мне убедительными, хотя я мало пока знаком с жизнью программ.

– Хорошие, понятные комментарии – это элемент не только культуры программирования, но и культуры общения, показатель отношения автора программы к коллегам и потенциальным потребителям программы в целом.

Чтобы освоить этот оператор, внесем еще одну строчку в нашу программу:

1 REM Пробная программа

Сказать по правде, программа наша пока скучная. Ведет она себя просто, "как дважды два", информацию выдает скудную. Такую программу можно, конечно, выполнить много раз, но смысла в этом нет. До сей поры она носит чисто показательный характер, однако дальше мы попытаемся развить ее так, чтобы приблизить к реальным задачам и одновременно больше узнать о Бейсике.

Прежде всего улучшим "лицо программы": то, что выдается в процессе ее работы на экран дисплея. Поскольку воспринимает информацию с дисплея человек, она должна быть удобной для восприятия. Человек плохо воспринимает, например, числа, не сопровождаемые никаким пояснительным текстом. Выдавать на экран информацию в удобной для восприятия форме тоже элемент культуры программирования. Персональный компьютер обладает для этого достаточно хорошими средствами.

"Пояснение" к цифре 4, выдаваемой нашей программой, могло бы выглядеть так:

ДВА В КВАДРАТЕ = 4

Программа станет от этого если не более "умной", то хотя бы более "вежливой", дружелюбной. Чтобы напечатать текст, можно использовать все тот же оператор PRINT, но текст при этом нужно взять в кавычки.

– Этот будет другой оператор?

– Давайте попробуем. Под каким номером вы будете вводить его в программу?

– Он должен выполняться перед печатью четверки, но, видимо, после установки позиции. Например:

8 PRINT "ДВА В КВАДРАТЕ ="

А теперь действуй.

RUN

На экране возникло:

ДВА В КВАДРАТЕ =

4
Ok

– Не совсем то. Число почему-то оказалось в следующей, 11-й строке.

– Я умышленно не поправил вас, чтобы вы лучше усвоили, как работает оператор PRINT. В том виде, как вы написали, он после печати текста автоматически "переходит" в начало следующей строки экрана. Называется это "переводом строки".

– И поэтому очередной оператор PRINT печатает на строку ниже?

– Да, так составлена наша программа. Давайте посмотрим на нее.

```
LIST
1 REM Пробная программа
2 CLS
5 LOCATE 10, 40
8 PRINT "ДВА  В  КВАДРАТЕ ="
10 PRINT 2 * 2
20 END
Ok
```

– А, я понял. Перед первой печатью мы установили позицию, сделаем это и перед вторым оператором PRINT.

– Какие параметры вы зададите оператору LOCATE?

– Нам нужно напечатать число в той же строке и в столбце ... несколько правее текста. Точнее, на столько позиций, сколько в тексте символов:

```
9 LOCATE 10, 56
RUN
                        ДВА  В  КВАДРАТЕ = 4
Ok
```

– Результата мы добились, но все это как-то сложно для такой простой задачи. Выходит, программирование довольно утомительная штука...

– В данном случае есть более простое решение для той же задачи. Оператор PRINT не переводит строку, если он заканчивается "точкой с запятой". Можно написать

```
8 PRINT "ДВА  В  КВАДРАТЕ =";
```

Строку он не меняет, а столбец как раз устанавливает в позиции последнего напечатанного символа, так что следующая печать начнется ...

– ... с нужного места?

– Да, поэтому оператор в 9-й строчке излишен.

– А можно удалить оператор с нашей простыни?

– Разумеется. Если вы пишете номер строчки без оператора:

```
9
```

то эта строчка становится "пустой", т.е. стоявший в ней оператор

удаляется. Чтобы еще немного улучшить вид нашей программы, я открою другую особенность оператора PRINT. Он может содержать несколько аргументов, перечисляемых через ";", и тогда они печатаются на экране подряд.

– Значит, можно два оператора, 8-й и 10-й, записать одним махом:

```
8 PRINT "ДВА  В  КВАДРАТЕ ="; 2*2
```

– Не забудьте удалить оператор в 10-й строчке, раз вы объединили его с предыдущим.

– А если этого не сделать, то....

– Печать четверки произойдет дважды. Бейсик исполнит все, не задумываясь о смысле вашей программы. Итак,

```
10
LIST
1 REM Пробная программа
2 CLS
5 LOCATE 10, 40
8 PRINT "ДВА  В  КВАДРАТЕ  ="; 2 * 2
20 END
Ok
RUN
                        ДВА  В  КВАДРАТЕ = 4
Ok
```

– Кропотливая работа, я вам скажу!

– Лиха беда начало...

Поразмыслим, что произошло. Мы задали исполнителю, интерпретатору Бейсика, программу, где предписали ему вычисления (пускай пока и очень простые) и предложили показать результат в удобной для нас форме. Очень многие практические задачи именно так и формулируются. Однако мы потратили некоторые усилия на совершенно банальную программу. Предположим нам необходимо будет выполнить другие вычисления, что же, писать новую программу? Мыслимых задач невероятно много, даже если ограничиться только вычислениями. А программы будут в чем-то очень схожи.

Перейдем же от арифметики к алгебре. Слава богу, с "иксами" мы все знакомы.

Изменим 8-й оператор так:

```
8 PRINT "ДВА  В  КВАДРАТЕ ="; X * X
```

– Не нужно ли и текст изменить: "X В КВАДРАТЕ"?

– Давайте:

```
8 PRINT "X В  КВАДРАТЕ = "; X * X
```

И что теперь, по-вашему, произойдет?

— Программа стала намного интереснее — теперь она вычисляет и печатает квадрат л ю б о г о числа, не так ли?

Проверьте!

RUN

X В КВАДРАТЕ = 0

Ok

— Да, результат получился еще более скучный. И непонятный.

— Вы хотите напечатать квадрат числа, но умалчиваете, какого именно. Что же делать Бейсику? В этом случае (так и говорят "по умолчанию") Бейсик считает, что X равен 0.

— Ну, хорошо, пусть теперь X равен 5.

— Обратите внимание на вашу фразу! Вы должны сообщить интерпретатору то, что вы только что сказали. В Бейсике есть оператор "пусть", и ваша фраза запишется так: LET X = 5. Где мы поместим его на нашем полотне?

— Взглянем-ка еще раз на программу:

```
LIST
1 REM Пробная программа
2 CLS
5 LOCATE 10, 40
8 PRINT "X В  КВАДРАТЕ ="; X * X
20 END
Ok
```

Попробуем действовать за интерпретатор. Я гашу экран, устанавливаю место, куда я начну печатать, печатаю ... Стоп! В этот момент я уже должен "знать", каков у меня X. Значит, нужно задать X перед тем. Где именно, по-моему, неважно. Например,

```
6 LET X = 5
```

— Исполняем?

RUN

X В КВАДРАТЕ = 25

Ok

— Странно. Результат-то правильный, но этот X... Мы получили что-то вроде уравнения?

— Если хотите, можете так истолковать. Но Бейсик лишь выполнил то, что предписано, не больше, не меньше. Посмотрим на 8-й оператор. Приказ LIST может печатать отдельные операторы, если указать номер оператора в качестве аргумента:

```
LIST 8
8 PRINT "X В КВАДРАТЕ ="; X * X
Ok
```

Оператором PRINT все, что в кавычках, печатается буквально, буква в букву, это ведь для него просто текст. Видите, вы сами задаете печать икса в первом аргументе оператора. Если кавычек нет, то аргумент сначала вычисляется, а потом уж печатается; потому-то мы и видим при исполнении результат умножения. Следовательно, икс останется иксом, пока он в кавычках. А если его вынести за кавычки и сделать еще одним аргументом оператора, он будет вычисляться:

```
8 PRINT X; "В КВАДРАТЕ ="; X * X
```

– Тут и вычислять-то нечего, взять X и все!

– Это тоже входит в понятие вычисления. Сейчас вы упомянули очень важную вещь. Наш X является *переменной*, а в программировании переменные играют не меньшую роль, чем в математике. "Взять X" значит взять *значение* переменной, *имя* которой X (значение равно в данный момент 5). Оператор LET придает значение переменной, или, как говорят, присваивает значение. Поэтому его называют оператором *присваивания*. В случае, когда мы вычисляем *арифметическое выражение* (например, "X в квадрате"), то сначала берутся значения входящих в него переменных (в данном случае X), а затем уже над значениями производятся арифметические действия ("пятью пять"):

```
RUN
                5 В КВАДРАТЕ = 25
Ok
```

– Хорошо. Воспользуемся нашим переходом к переменным. Ведь X теперь можно задать любой. Проверим наш старый вариант с двойкой:

```
6 LET X = 2
RUN
                2 В КВАДРАТЕ = 4
Ok
```

Мы сделали важный шаг. Наша программа путем простого изменения стала намного универсальнее. Теперь она действительно может возвести в квадрат любое число, не только целое. Такое действие уже не всегда выполнишь в уме. Для этого требуется изменить только одну строчку: присвоить X новое значение. Например, действительное число:

```
6 LET X = 1.8972
RUN
                1.8972 В КВАДРАТЕ = 3.599368
Ok
```

С переменными мы сталкиваемся не только в математике, но и в обиходе. Простейшие переменные связаны с величиной, количеством и принимают числовые значения. Хозяйка, у которой на кухне хранятся по сосудам и сусекам, баночкам и коробочкам крупы и чай, варенье и масло, постоянно оперирует переменными величинами. Надпись "чай" на жестяной коробочке — это имя переменной; ее содержимое — значение переменной. Когда коробочка пуста, то в наших терминах значение переменной "чай" равно нулю.

Наоборот, переменную в программе можно представлять себе как своеобразную коробочку, на которую наклеен ярлык — имя переменной. Когда мы хотим взять значение переменной, мы смотрим на содержимое коробочки. Меняя значение, мы кладем в коробочку что-то новое. (Про переменную иногда говорят, что в нее "засылается значение".)

Многие действия, на кухне ли, в компьютере, используют только ярлыки-имена, а не конкретные значения. Заварка чая — достаточно хорошо определенная процедура, и если нас попросят заварить побольше, то такая модификация нас не затруднит. Она имеет чисто количественный характер и не влияет на последовательность наших действий. Точно так же оператор PRINT в нашей программе делает свое дело независимо от того, чему в этот момент равно значение X. В таких случаях переменную называют параметром той программы или процедуры, в которой она используется. Параметров в программе может быть, конечно, несколько.

Посмотрим, как можно видоизменить нашу программу путем введения параметров. Но прежде вернемся на минуту к нумерации операторов. Как видим, сами номера никак интерпретатором не используются, важна лишь последовательность их расположения на простыне. Мы "сажали" их на простыню не подряд, а с "дырами", и потом пользовались этим для изменения программы, в частности для вставки других операторов в свободные строчки. Ясно, что вставлять беспредельно нам не удастся, так как все строчки будут заняты и дыр не останется.

Чтобы это не служило препятствием для составления программы, в Бейсике имеется оператор RENUM, который перенумеровывает строчки на простыне. Применим его к нашей программе и посмотрим на результат:

```
RENUM
Ok
LIST
10 REM Пробная программа
20 CLS
30 LOCATE 10, 40
```

```
40 LET X = 5
50 PRINT X; "В КВАДРАТЕ ="; X * X
60 END
Ok
```

После перенумерации операторы программы расположились с интервалом в 10 строчек, а порядок их сохранился. Появились достаточные для вставки новых операторов "дыры".

Программа в измененном виде такова (прикиньте сами, какими манипуляциями были достигнуты эти изменения):

```
LIST
10 REM Пробная программа
20 CLS
30 X = 5: L = 10: C = 40
40 LOCATE L, C
50 PRINT X; "В КВАДРАТЕ ="; X * X
60 END
Ok
```

Оператор LOCATE в качестве аргументов теперь использует вместо чисел переменные, которые задают строку и столбец экрана и названы соответственно L и C.

Присваивание значений всем параметрам происходит в одной строчке, до операторов, их использующих. При этом используется составной оператор из трех операторов LET, записанных в сокращенной форме (оставлен только знак равенства между переменной и значением, а имя оператора опущено).

Легко видеть преимущество такой формы программы в сравнении с первоначальной. Печать квадрата л ю б о г о числа в л ю б о й строке и л ю б о м столбце экрана достигается с помощью такой программы изменениями, вносимыми только в одну, 10-ю строчку – в один из операторов присваивания.

И переменные вообще, и параметры в частности, могут принимать не только числовые значения. Другим т и п о м значений (мы это видим из работы оператора PRINT) являются строки символов, заключаемые в кавычки. В Бейсике, как и в большинстве других языков программирования, принято, чтобы переменная принимала значения только одного типа: в баночку для чая вы не кладете масло. Переменную, конечно, не нужно "отмывать" от старого значения, но транслятору обычно проще и удобнее, чтобы переменная принадлежала к конкретному типу, в случае Бейсика – целому, вещественному или символьному (строковому). Заметим, что есть строки, внешне похожие на числа, например "59", но обычные числовые операции с ними невозможны.

Чтобы дать понять интерпретатору, каков тип переменной, используется последний символ в имени переменной:

для целочисленных переменных это % (процент); например: N %

для вещественных – ! например: TOTAL!

для символьных – $ (знак доллара); например: MESSAGE $

"По умолчанию", если последний символ не есть ни один из трех указанных, переменная считается числовой, т.е. принимает как целые, так и вещественные значения (но не символьные). Этим мы и пользовались до сих пор в нашей программе. Однако следующий оператор

HELLO = "Здравствуйте!"

повлечет сообщение об ошибке: НЕСООТВЕТСТВИЕ ТИПА, а правильным будет:

HELLO $= "Здравствуйте!"

Параметры, которыми оперирует программа, часто объединяются общим термином д а н н ы е. Для нашей программы данными являются значения, определенные в 30-й строчке. Многие практические задачи можно сформулировать как задачи переработки данных. В Бейсике есть еще один способ работы с данными, который мы продемонстрируем, изменив соответствующим образом программу:

```
15 DATA 5, 10, 40
30 READ X, L, C
```

Значения, которые мы собираемся использовать, сгруппированы в операторе DATA и перечислены через запятую. Можно говорить, что оператор DATA просто хранит значения, потому что при его исполнении ничего не происходит. Таких операторов и хранимых в них значений может быть в программе много; при этом важен лишь общий порядок, в котором следуют операторы, и порядок перечисления значений внутри оператора. Процедура READ "считывает" данные из оператора DATA и присваивает их переменным – точно так же, как это было в предыдущем варианте программы. Типы переменных и значений должны совпадать, иначе фиксируется ошибка.

Чтобы полностью понять работу операторов DATA и READ, достаточно представлять, что все данные во всех операторах DATA занесены подряд на отдельную простыню, а оператор READ продвигается по ней, считывая в каждую свою переменную одно значение. Такая форма работы с данными более удобна, если мы имеем большие массивы однородных величин.

– В одном из самых первых разговоров вы упомянули о программах, ведущих диалог с человеком, однако до сих пор наша связь с программой была односторонней. Программа нам что-то сообщала, и только.

– Что ж, пойдем дальше. В каждом языке программирования имеются средства, позволяющие организовать диалог программы с тем, кто ей пользуется (так называемым пользователем). Форму диалога вы, программист, можете выбрать на свое усмотрение.

Простейший диалог можно организовать в Бейсике с помощью оператора INPUT:

```
LIST
10 REM Пробная программа
20 PRINT "Как вас зовут?";
30 INPUT  NN$
40 PRINT  "Здравствуйте, "; NN$
50 INPUT "Какое число вы хотите возвести
в квадрат?", X
60 INPUT "В каком месте экрана показать результат"; L, C
70 CLS
80 LOCATE  L, C
90 PRINT  X; "В КВАДРАТЕ = "; X * X
100 END
Ok
```

Одно из неудобств, с которым мы сталкивались в нашей первой программе, заключается в том, что выполнить программу с новыми параметрами можно было только путем ее изменения, пусть даже в одной строчке. С помощью оператора INPUT этого можно избежать. В новом варианте программы никакие конкретные значения не фигурируют, они поступают от пользователя в ходе ее исполнения. Его ответы и определяют различные варианты работы программы.

Исполним:

```
RUN
Как вас зовут? Петя
Здравствуйте, Петя
Какое число вы хотите возвести в квадрат? 1.73
В каком месте экрана показать результат? 2,1
```

Происходит гашение экрана и печатается:

```
1.73 В КВАДРАТЕ = 2.9929
Ok
```

При исполнении оператора INPUT программа приостанавливается – она ожидает вашего ответа. Вы набираете на клавиатуре свой ответ, завершая его клавишей ввода, так же, как набираете текст приказа или оператора. Ваш ответ – в первом случае это строка символов П + е + т + я – изображается на экране; кавычки при этом необязательны.

Ваш ответ запоминается в переменной, которая является аргументом оператора. В строчке 30 это символьная переменная NN$.

(Обратите внимание, что тип переменной соответствует введенному вами значению.) Следующий оператор, PRINT, печатает приветствие, используя только что введённое значение. Как видите, в данном случае программа прибегает к диалогу, чтобы узнать от вас то, что она не может получить ни из какого другого источника.

Обратим внимание на различные формы оператора INPUT. В строчках 50 и 60 он снабжен дополнительным аргументом в виде строки; этот аргумент печатается до ввода значений.

В операторе 60 первый аргумент отделен от второго точкой с запятой, а не запятой, как в предыдущем случае. Это указание оператору автоматически завершить печать сообщения вопросительным знаком (поэтому знак вопроса не нужно включать в текст сообщения). В строчке 60 оператором INPUT запрашиваются сразу две переменные, заданные через запятую. При вводе ответа пользователь должен задать также два значения через запятую.

Оператор INPUT дает нам новый способ работы с данными в программе. В отличие от ранее рассмотренных (присваивание, операторы DATA–READ) этот способ не требует, чтобы данные заранее, еще до исполнения, вносились в текст программы. Вместе с тем возможность запрашивать информацию у пользователя отнюдь не означает, что этот способ манипулирования данными должен преобладать над другими. То, что иногда кажется гибким решением программисту, с точки зрения пользователя может стать обузой. Если заставлять пользователя отвечать на нерезонные, неуместные на его взгляд вопросы, то можно этим вызвать его раздражение, и он предпочтет вашей программе другую.

Подавляющее большинство программ, используемых на персональных ЭВМ, имеет диалоговый характер, т.е. предполагает взаимодействие человека с программой в ходе ее исполнения.

Задачи, решаемые программами, могут быть самые разные: вычислительные, связанные с обработкой текста, с хранением и поиском информации, задачи обучения человека. Некоторые интересные виды задач рассмотрены в статье о прикладных программах.

Кстати говоря, мы постоянно имеем дело с программой, ведущей с нами активный диалог. Это интерпретатор Бейсика.

– В чем же здесь диалог?

– Кое-что вы ему сообщаете, кое-что он вам. Вы даете ему приказы, он их исполняет и отвечает "Ok".

– По-моему, он только слушает, что я говорю.. А единственная реплика "Ok" на человеческом языке значит просто "Что прикажете?" Разве это диалог?

– Бейсик и еще кое-что сообщает, например об ошибках в вашей программе. Можете называть это и как-нибудь по-другому, если слово диалог вам кажется натяжкой.

Во многих практических случаях, действительно, диалог принимает такую форму, которую уместнее называть сотрудничеством человека и программы. Программа становится инструментом для решения задач, подобно другим окружающим нас средствам: телефону, автомобилю. Нажимая педали, кнопки, вращая диск, мы заставляем инструмент работать на нас. Составляя программу, мы можем предусмотреть в ней любой обмен информацией с человеком, иначе говоря, "встроить" в нее те же "кнопки" и определить, какой они будут нести смысл для пользователя, и значит, как их нужно обрабатывать в программе. На примере Бейсика мы видим, что программа может помогать в составлении других программ.

– Я все хочу спросить, что за манипуляции вы выполняете в начале каждого сеанса работы?

– Мне не хотелось раньше времени обременять вас деталями, но теперь я объясню свои действия. Прежде всего,где находится наша Бейсик-программа в процессе работы на машине?

– Мы об этом говорили. Программа хранится на диске в одном из файлов, но при исполнении загружается в оперативную память.

– Это верно. Пока программа находится на диске во в н е ш н е й памяти, ее нельзя ни изменить, ни исполнить. Такова архитектура компьютера, и основное соображение в пользу этого – скорость работы. Постоянно обращаться к диску было бы слишком накладно. Программа сначала переносится в о п е р а т и в н у ю, быструю память машины, а уж затем исполняется или подвергается изменениям.

– А что было, когда мы только начали создавать программу?

– Мы создали ее в оперативной памяти, а потом перенесли в дисковый файл.

– Если можно держать программу в оперативной памяти, то зачем вообще использовать диск?

– Во-первых, оперативная память сравнительно невелика, а исполнять нужно много разных программ. Все они туда все равно не поместятся. Во-вторых, содержимое оперативной памяти теряется при выключении машины в отличие от внешней памяти, которая "постоянна".

– Значит, мы записываем на диск, чтобы "не забыть", а потом считываем, чтобы "вспомнить"? Как же это делается в Бейсике – есть специальные операторы?

– Оператор SAVE сохраняет программу в файле с указанным именем, имя файла берется в кавычки:

```
SAVE "MYPROG"
```

Чтобы произвести обратную операцию, надо обратиться к оператору LOAD:

```
LOAD "MYPROG"
```

После этого программа считывается из файла в оперативную память.

– Понятно. Именно эти операции вы и производите перед выключением машины и после ее включения.

– Да, но это не все. После включения машины я попадаю не сразу в Бейсик, а в операционную систему. Но это уже отдельный разговор. Кое-что об операционных системах можно узнать из предыдущей статьи.

3. НОВЫЕ ГОРИЗОНТЫ

Вскользь мы уже говорили, что программы – это достаточно сложно построенные последовательности действий. Упоминали мы о влиянии "обстановки" на работу программы, подчеркивая, что в ней могут учитываться различные условия.

Что же такое обстановка? Какие условия можно проверять в программе и как? В реальной жизни условия фигурируют в наших планах в виде различных "если". "Если магазин открыт, то все в порядке ..., иначе (т.е. если закрыт) то-то и то-то. Стоп, а денег хватит? (т.е. если не хватит, то ...)". Проверяемые условия: "состояние магазина", "наличие денег". В Бейсике имеется оператор "если", который управляет выполнением других операторов также в зависимости от некоторого условия, поэтому он называется у с л о в н ы м оператором. Условия в Бейсике имеют обычно вид сравнений: "равно", "не равно", "больше", "меньше".

– А что при этом сравнивается?

– Сравнивать два конкретных числа бессмысленно – результат сравнения известен заранее. Сравнивать надо вещи, которые до исполнения программы, во время ее составления неизвестны, неопределенны, могут меняться.

– Вы клоните к переменным?

– Совершенно верно. Переменная может принять различные значения, и нам может быть отнюдь не безразлично, какое именно. Примеры условий: "X равен 5", "L больше нуля".

– А "магазин закрыт"?

– Магазин в данной ситуации имеет для нас один важный признак: "открытости". "Магазин закрыт" – констатация отсутствия признака. В программе это интерпретируется так: переменная "магазин" принимает одно из двух значений – 0 или 1. Поскольку эти значения можно интерпретировать как "ложь–истина", то такая переменная называется л о г и ч е с к о й. Следовательно, ваше условие означает то же, что "X равен нулю", или "X есть ложь". Переменные могут сравниваться и друг с другом: "A больше B", "денег больше, чем стоит буханка".

– Какой же вид имеет условный оператор?

– Смотрите, я добавляю его в нашу программу:

25 IF X = 5 THEN PRINT "Люблю пятерку!"

После слова IF следует условие, затем слово THEN и оператор, исполняемый, когда условие выполнено.

– Позвольте, условие пишется так же, как присваивание?

– Да, это может сначала вызвать неудобство, но вообще-то их трудно спутать: условие всегда следует за словом IF, присваивание может следовать только за словом THEN.

– Итак, печать текста о любви к пятерке будет происходить только при X, равном 5. Проверим. Неясно пока, в какой позиции экрана будет печататься этот текст.

– После оператора CLS текущей позицией для печати является левый верхний угол экрана: первая строка, первый столбец.

– Тогда давайте установим какую-нибудь позицию, например на строку ниже в том же столбце.

23 LOCATE 11, 40

– Почему вы задали такие значения аргументов?

– Но ведь основная печать у нас в 10-й строке в 40-м столбце. Это видно по значениям параметров L и C.

– А если мы изменим эти значения, например, 16 на 30?

– Тогда... Тогда придется изменить и мою 23-ю строчку. Да, это неудобно.

– Я вам столько говорил про параметры, а вы ... ничего не усвоили.

– Понял, понял. Нужно использовать переменные. На строку ниже – значит, на единицу больше, чем параметр L. В том же столбце – значит... Ну, ясно.

23 LOCATE L + 1, C

– Лучше, но еще не все.

– По-моему, все нормально. Давайте запустим?

```
RUN
                    5 В КВАДРАТЕ = 25
                     Люблю пятерку!
Ok
```

Отлично. Интересно, работает ли условие и как программа вообще реагирует на другие значения параметров?

```
20 X = 32: L = 16:C = 30
RUN
                32 В КВАДРАТЕ = 1024
Ok
```

— Работает программа правильно, но все же в ней есть один изъян. Посмотрите:

```
LIST
10 CLS
20 X = 32: L = 16: C = 30
23 LOCATE L + 1, C
25 IF X = 5 THEN PRINT "Люблю пятерку!"
30 LOCATE L, C
40 PRINT X; "В КВАДРАТЕ =" ; X * X
50 END
Ok
```

Говоря профессиональным языком, она не совсем оптимальна. Зачем нужен оператор в 23-й строчке?

— Он устанавливает позицию для последующей печати в строчке 25.

— А если ее не произойдет, этой печати?

— Ах да, она же "при условии": если X не равен 5, то печати про любовь не будет. Тогда и первый оператор LOCATE, который только для этой печати и требуется, тоже не нужен.

— Он тоже выполняется условно. А вы заставляете интерпретатор выполнять его в любом случае. Он напрасно тратит на это время. Он же такой исполнительный, он делает все, что просят. Вам не жалко его зря гонять?

— Вы выбираете чисто человеческую аргументацию. А сами говорили, что это всего лишь "винтики", "железо"!

— Программист вообще-то склонен одушевлять свою программу. Он привыкает думать и говорить, что это программа "работает", а не транслятор и не железо. И знаете, многие программы этого заслуживают. Но заботясь об оптимальности, или, как еще говорят, об эффективности, программист заботится и о том, кто будет пользоваться программой. Если она выполняет лишние операторы, то тратит время потребителя, и эти затраты могут быть далеко не так безобидны, хотя в нашем случае оператор LOCATE и выполнится очень быстро.

— Давайте и его поставим под условие:

```
23 IF X = 5 THEN LOCATE L + 1, C
```

— Снова неоптимально! Вы два раза подряд, в 23-м и 25-м операторах, проверяете одно и то же условие. Вернее, заставляете делать это Бейсик. Он у вас будет твердить, как рассеянный человек: "X равен 5 или нет? Равен или нет?" Это повод призадуматься: нельзя ли и здесь сэкономить?

— Проверить условие один раз и сразу выполнить два оператора? Объединим их в составной, как уже делали:

```
25 IF X = 5 THEN LOCATE L + 1, C: PRINT "Люблю пятерку!"
```

И не забыть стереть 23-й, ставший лишним:

```
23
LIST
10 CLS
20 X = 32: L = 16: C = 30
25 IF X = 5 THEN LOCATE L + 1, C: PRINT "Люблю пятерку!"
30 LOCATE L, C
40 PRINT X; "В КВАДРАТЕ ="; X * X
50 END
Ok
```

– Я объяснил вам условный оператор в его краткой форме. В полном виде он выглядит так: "если ... то ... иначе ...", т.е. позволяет указывать, что делать при выполненном условии и что – в противном случае, например,

```
25 IF X = 5 THEN LOCATE L + 1, C: PRINT "Люблю пятерку!"
   ELSE
   LOCATE L + 1, C: PRINT "Это не пятерка"
```

А теперь я у вас снова спрошу, в чем неоптимальность программы?

– Мы два раза выполняем в 25-й строчке одно действие: LOCATE L + 1, C.

– Однако мы не делаем лишних действий ни при X = 5, ни в противном случае. Дело тут в том, что хотя в ы п о л н я е т с я каждый раз лишь один оператор LOCATE, з а п и с а н он дважды. И этого можно избежать. Проверка условия происходит слишком рано, ее можно делать уже после оператора LOCATE, который нужно выполнять в любом случае. Оператор можно вынести из условного оператора, как бы за скобки:

```
25 LOCATE L+1, C: IF X=5 THEN PRINT "Люблю пятерку!"
   ELSE PRINT "Это не пятерка"
```

Программа стала короче, хотя она делает то же самое.

Мы столкнулись с двумя основными соображениями эффективности в программировании: скорость исполнения программы и занимаемый объем, или "память". При прочих равных условиях программист стремится добиться максимума первого и минимума второго. Часто, однако, эти требования становятся противоречивыми и одно может быть достигнуто только за счет другого. В программировании действует что-то вроде "золотого правила" механики: "выигрываешь по скорости – проигрываешь по памяти.

Пока мы не рассматривали условий, все операторы, нанесенные нами на простыню, обязательно исполнялись. Интерпретатор дви-

гался по ней только вперед и рано или поздно доходил до каждого оператора. С введением условного оператора это правило впервые нарушилось. Внутрь условного заключены другие операторы, которые выполняются не всегда.

Условный оператор – это развилка. Выбирая один вариант, транслятор отвергает другой, руководствуясь при этом строгим критерием: выполнением – невыполнением условия. В том виде, однако, как мы его использовали, возможности разветвления ограничены. Длина любого оператора в Бейсике имеет свой предел, хотя он и может занимать более одной строки экрана, поэтому вставить достаточно много операторов внутрь условного оператора нам не удастся.

Для удобства организации развилок в программах Бейсик имеет специальный оператор, который меняет порядок исполнения других операторов. Это оператор перехода, GOTO, который указывает номер, с которого нужно продолжать исполнение программы, например,

```
100 GOTO 200
```

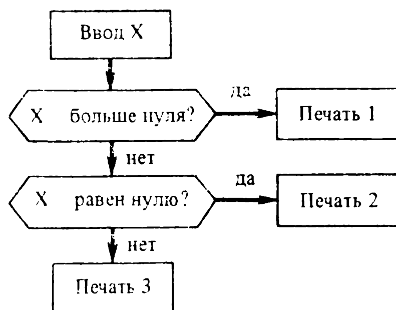
Пользуясь возможностью, которую обеспечивает нам оператор перехода, мы можем выписать где-нибудь отдельно целый кусок программы и указать переход на него, или, как говорят, передать управление. Проанализируйте, например, следующий фрагмент программы:

```
10 INPUT "Задайте число:", X
20 IF X>0 THEN GOTO 70
30 REM Сюда мы никогда не попадем при положительном X
40 IF X=0 THEN PRINT "Нуль" ELSE PRINT X; "– отрицательное число"
50 REM Нужно обойти оператор 70, исполняемый только для положительного X
60 GOTO 80
70 PRINT X; "– положительное число"
80 END
```

Благодаря переходам исполнение программы становится сложным, нелинейным, однако Бейсик устроен так, что все операторы независимо от условий их исполнения помещаются на одно общее полотно, т.е. запись программы остается линейной. В связи с этим нужна аккуратность "вывешивания знаков объезда" – операторов перехода – на такой дороге.

В вышеприведенной программе после оператора 20 возникают два альтернативных куска программы (операторы 30–60 для неположительных X и оператор 70 для положительных). Оператор 60 есть необходимая предосторожность: обходится оператор другой альтернативы. Для отражения логики исполнения программы, или,

что то же самое, ее структуры управления, применяется условная форма записи программы, называемая блок-схемой, или структурной схемой. Блок-схема нужна не для транслятора, а для человека, и часто является начальным этапом составления программы. На рисунке представлена блок-схема для нашей программы.



— Зададимся следующей задачей: напечатать таблицу умножения на 9. Средства ее решения не выходят за пределы того, что нам известно.

— Одно решение сразу приходит в голову. Поскольку такую таблицу можно рассматривать как текст, набор знаков, то напечатать этот текст можно с помощью нескольких операторов PRINT. Например, PRINT "2*3=6" Но я чувствую, что вы хотите чего-то другого.

— Задумаемся вот о чем. Какова может быть длина программы? Ваше решение потребует, скажем, 9 операторов. А если печатать всю таблицу умножения? Вообще, если каждое фактически исполняемое в программе действие записывать отдельным оператором, то при больших расчетных задачах требовалось бы огромное время на то, чтобы з а п и с а т ь программу. И объем такой программы был бы колоссальный.

— Выход один — повторять исполнение одних и тех же операторов много раз. Благодаря оператору перехода мы осознали, что то, как программа записана, и то, как она исполняется, это разные вещи. Но повторять совершенно одно и то же бессмысленно.

— Оператор может быть один, но его действие — переменным, если...

— Если он использует переменные в качестве своих аргументов!

— Вернемся к таблице умножения на 9. Мы девять раз должны сделать почти одно и то же. Вернее одно и то же, но...

— ... с разными параметрами? Я попробую написать! Основное действие — умножение и печать результата:

```
10 PRINT X; "*9="; X*9
```

Начинаем с единицы:

```
5 X=1
```

Теперь нужно взять следующий X и снова выполнить 10-й оператор. В этом вся соль. Чтобы взять следующий, нужно X увеличить на 1. А как?

— Давайте я помогу:

```
20 X=X+1
```

— Это выглядит так, как будто X равен самому себе плюс единица.

— Не смотрите на это глазами математика. Это вовсе не равенство, а присваивание. Слева стоит переменная, с которой мы имеем дело, X. Справа стоит з н а ч е н и е, которое попадает в эту переменную; оно складывается из прежнего значения X и единицы. Вы делаете то же самое, когда досыпаете в баночку чай – что-то там уже есть. Это как раз предыдущее значение нашей переменной.

— Все-таки этот знак равенства может ввести в заблуждение.

— Вы правы. Видимо, поэтому в других языках программирования для операции присваивания используется иное обозначение. Например, в языках Алгол и Паскаль последний оператор выглядел бы как X:=X+1.

— Заключительный шаг очевиден:

```
30 GOTO 10
```

— Прежде чем исполнить программу, взгляните на нее:

```
LIST
5 X=1
10 PRINT  X;"*9 = "; X*9
20 X=X+1
30 GOTO 10
Ok
```

и ответьте, когда она должна закончиться?

— Так... печатаю, прибавляю, перехожу, снова печатаю. И так далее. Похоже, что никогда не кончится. Это ошибка. Будут печататься все целые числа, умноженные на 9, а нам нужны только из первого десятка. Значит, нужно вовремя остановиться. Как только X станет больше 9... Сейчас изменим:

```
25 IF X>9 THEN  GOTO  40
40 END
```

— Вот теперь ваша программа сработает правильно. То, что вы сейчас делали, называется организацией цикла, поскольку серия операций производится циклически, несколько раз.

Ситуация "бесконечного" исполнения программы, с которой мы столкнулись, называется у программистов з а ц и к л и в а н и е м, а ваше исправление – выходом из цикла. Организация цикла требует осторожности. Если ваша программа долго работает без каких-то видимых результатов, то очень возможно, что вы забыли указать условие выхода из цикла.

Циклы приходится организовывать очень часто, всякий раз писать однотипные последовательности действий было бы утомитель-

но и чревато ошибками, поэтому в Бейсике (и других языках программирования) есть средства для быстрого описания циклических операций. Это операторы ц и к л а. Вот как с помощью такого оператора записывается наша программа:

```
10 FOR X=1 TO 9 STEP 1
20 PRINT X; "* 9 ="; X*9
30 NEXT X
40 END
```

Оператор цикла состоит из нескольких частей, расположенных в разных строчках: FOR ... TO ... STEP ... NEXT ..., которые указывают соответственно п е р е м е н н у ю ц и к л а и ее начальное значение (X=1), конечное значение (9), ш а г – значение, добавляемое к переменной цикла при каждой и т е р а ц и и – повторении (если шаг равен 1, как в данном случае, то его можно не указывать).

Оператор PRINT составляет т е л о цикла, т.е. те действия, которые повторяются. Он располагается в следующей строчке; таких операторов может быть несколько. В отдельной строчке располагается слово NEXT, за которым следует переменная цикла (ее можно опускать). Это сигнал о том, что тело цикла закончено и можно переходить к следующей итерации. Хотя оператор FOR составляет единое логическое целое, отдельные его части (FOR...TO...STEP и NEXT) ведут себя как самостоятельные операторы в том отношении, что могут входить в составной оператор, например:

```
10 FOR X=1 TO 9 STEP 1: PRINT X; "* 9 = "; X*9: NEXT X
```

(Такой оператор эквивалентен трем операторам в строках 10–30).

Пользоваться оператором цикла удобнее и надежнее, чем писать цикл вручную, как это мы с вами сделали вначале, – хотя бы потому, что ошибка зацикливания исключена. Вы просто вынуждены указать конечное значение переменной цикла, иначе вы не запишете оператор.

4. НОВЫЙ ИСПОЛНИТЕЛЬ

Чтобы получить более полное представление о языках программирования, мы познакомимся с новым исполнителем – интерпретатором языка Лого – и посмотрим на его отличия от Бейсика. Полем нашей деятельности мы изберем графику, вещь очень характерную для персонального компьютера. В предыдущей статье книги вы получили достаточно сведений о цветных графических мониторах, которые являются устройством отображения информации на персональных компьютерах.

Экран монитора устроен по такому принципу, который нам,

свидетелям грандиозных массовых праздников в Лужниках, усвоить будет нетрудно. Вообразите трибуну в 200 рядов по 320 мест в ряду, где на каждом месте сидит человек, вооруженный несколькими цветными дощечками. Одну из них он всегда держит над собой. При взгляде издалека эта мозаика дощечек дает достаточно детальное цветное изображение, вполне похожее на телевизионное. Каждая точка на экране – это человек с дощечкой.

– В Бейсике, видимо, есть особые операторы для управления цветом точек на экране?

– Да, мы рассмотрим некоторые из них. Но сначала вспомним декартовы координаты: это существенно для использования графики из Бейсика. Начало координат располагается в левом верхнем углу, а ось Y ориентирована не вверх, а вниз – так, что точки экрана нумеруются в том же порядке, что знакоместа, только их больше. В нашем распоряжении часть плоскости, причем X меняется от 0 до 319, а Y – от 0 до 199.

– Это соответственно место и ряд?

– Да, и мы будем управлять сидящими на этих местах людьми. Впрочем, одно предварительное действие. Нужно сказать Бейсику, что экран мы используем графический – с помощью оператора SCREEN:

```
SCREEN 1
Ok
```

Перед нами черный экран. Теперь смотрите, я говорю:

```
PSET (160, 100), 1
Ok
```

и человек в 100-м ряду на 160-м месте поднимает зеленую дощечку.

– Единичка обозначает зеленый цвет? А какие еще у него дощечки?

– Всего четыре: 0 – черная (эту он держит вначале), 1 – зеленая, 2 – красная, 3 – белая.

– Вашу точку еле видно.

– А можно ли различить одного человека на гигантском стадионе? Вот когда они действуют согласованно, тогда будет другой эффект. Например, мы можем изобразить отрезок:

```
LINE (160, 100)–(80, 150), 2
Ok
```

Этот отрезок соединяет две точки, с указанными мной координатами. Он красного цвета и равен четверти диагонали экрана. Разумеется, линии на экране не абсолютно гладкие, поскольку складываются из точек.

– А окружность – тоже можно нарисовать?

– Оператор окружности выглядит так:

CIRCLE (160, 100), 80,,,,1

Ok

– Снова зеленый цвет. А для чего так много запятых?

– Это пропущенные аргументы. Дело в том, что данный оператор может рисовать не только окружность, но и эллипсы, причем не полностью, а лишь отдельные дуги. При этом используются опущенные нами аргументы.

– Мы все время должны задавать координаты. Но это взгляд инженера или математика. Ребенок ведь не знает декартовой плоскости, но и у него неплохо получается.

– Совершенная правда. Бейсик предлагает нам определенный взгляд на процесс рисования, и он не всегда удобен. Попробуйте, например, нарисовать снежинку.

– Можно, конечно, но придется повозиться с координатами.

– А вот Лого действует по-другому, он встает на точку зрения ребенка. Ребенок все время решает задачу: как переместиться туда-то, если сейчас я здесь? Такое перемещение по плоскости относительно.

– Да он порой не отрывает карандаша от бумаги.

– Лого предлагает ребенку карандаш. Он говорит: "В центре экрана сидит черепашка с пером в руке. Ты можешь командовать: вперед, назад, поворот". Черепашка исполняет и тем самым рисует.

– Следовательно, есть операторы "вперед", "назад"?

– Пока мы беседовали, я "вызвал" интерпретатор Лого и он будет исполнять, что прикажете.

– Этот маленький треугольник в центре и есть черепашка? Хочу скомандовать "вперед!".

– Пожалуйста. Но нужно сказать, на сколько вперед:

FORWARD 50

– Движение измеряем в точках? А если в сторону?

– Лого сам выбирает масштаб. Но вы, по существу, упомянули новый оператор:

RIGHT 90

– Черепашка повернулась направо! Причем на прямой угол, следовательно, поворот измеряется в градусах.

– А вот команда "назад":

BACK 30

– Нарисуем теперь квадрат. Как очистить экран для новых упражнений?

– Просто скажите, что начинаете рисовать:

DRAW

– Квадрат – это четыре раза вперед с четырьмя поворотами. Размер стороны сделаем равным, например 40:

```
FORWARD 40
RIGHT 90
FORWARD 40
RIGHT 90
FORWARD 40
RIGHT 90
FORWARD 40
RIGHT 90
```

– Смотрите, вы повторяете одни и те же действия. В Лого можно записать повторение короче:

```
REPEAT 4 [FD 40 RT 90]
```

т.е. "повторить четырежды", а что повторить, я указываю в квадратных скобках. Запись операторов сокращенная, из 2-х букв. В Лого это допустимо.

– В Бейсике для повторения мы бы использовали оператор цикла. Но здесь запись проще, нет переменной цикла. Интересно, почему мы обошлись без нее?

– Когда мы имели дело с Бейсиком, переменная цикла играла роль параметра в теле цикла, в операторе PRINT, помните? Вам же параметр не потребовался, вы повторяете буквально одно и то же.

– Но результат разный. Как это объяснить?

– Вас не удивляет, что "два шага вперед" приводит к разным результатам?

– ... в зависимости от того, где я стою. Я понял: дело в том, что команды Лого относительны. Это все и упрощает. Скажите, Лого все время рисует желтым, но я надеюсь, можно задать и другой цвет?

– Конечно. Оператор SETPC устанавливает новый цвет пера черепашки. Например, красный:

```
SETPC 2
```

– А оторвать перо можно? Нельзя же все рисовать сплошной линией!

– И это черепашка умеет. PENUP – поднять перо (и значит, не рисовать, а только двигаться), PENDOWN – опустить и рисовать снова.

– Да, Лого совсем не похож на Бейсик.

– Вы еще лучше почувствуете это, если попробуете нарисовать окружность.

– В Лого есть такой оператор?

– А вы сами сообразите без чужой помощи.

– Простите, это легче сделать в Бейсике. Уравнение окружности я, пожалуй, припомню. Наставлю точек и получу окружность даже без помощи оператора CIRCLE.

– Разве вам не интересно сменить точку зрения – сесть на черепашку и начать командовать?

– Надо подумать. А пока объясните: как в Лого составляется программа? До сих пор мы исполняли лишь отдельные операторы.

– Здесь Лого отличается от Бейсика еще сильнее. В одном из наших самых первых разговоров мы обсуждали составление планов. И вам не понравилось...

– ... слишком детальное планирование: "спуститься по лестнице" и тому подобное.

– Не кажется ли вам, что Бейсик так и заставлял нас планировать?

– Бейсик ограничивал нас своими мелкими операторами? Пожалуй. Хотелось бы уметь создавать свои собственные, новые операторы.

– Заметьте, что вы требуете от языка программирования именно то, что так хорошо умеет человек: усваивать новые стереотипы и применять их в различных ситуациях. Человек упаковывает свои знания и вешает ярлычок. Ребенок узнает, как кипятить воду, составляя требуемое действие из более мелких, ему известных. После этого он получает новое средство, новую процедуру, которую использует в дальнейшей деятельности как целое: он это умеет.

В Лого вы можете создать новую процедуру (именно этот термин используется почти во всех языках программирования), или, что то же самое, новый оператор, и дать процедуре имя. Чтобы написать новую процедуру, вы пользуетесь особым оператором Лого TO (читать его можно: ДЛЯ):

```
TO BOX
REPEAT 4 [FD 40 RT 90]
END
```

То есть: ДЛЯ рисования квадрата сделай то-то до слова END. После этого вы получаете оператор BOX. Смотрите, оператор

```
BOX
```

дает нам тот же результат, что и прежде.

– Замечательно. Теперь можно развернуться!

– Не хотите к окружности вернуться? Я помогу. Будем развивать пример с квадратом. Попробуем нарисовать другие правильные многоугольники.

– Мне нравится шестиугольник. Так, что изменится? Поворот нужно делать на... кажется, на 60 градусов, и поворотов будет 6. Итак:

```
REPEAT 6 [FD 40 RT 60]
```

Получилось!

– А другие многоугольники?

– Попробуем. Все будет очень похоже. Рисование 4- и 6-угольников отличалось углом поворота и числом сторон. Причем угол поворота равен 360, деленному на число сторон.

– Или, другими словами, суммарный поворот равен 360 градусам. Это "закон черепашки", который действует для всех замкнутых фигур.

– Я, кажется, могу написать оператор рисования произвольного многоугольника, но для этого придется ввести параметр.

– Переменные в Лого должны начинаться с двоеточия.

– Тогда пишем:

```
REPEAT :SIDES [FD 40 RT 360/:SIDES]
```

– Ошибка. SIDES НЕ ИМЕЕТ ЗНАЧЕНИЯ. Это та же ошибка умолчания, которую вы когда-то допустили в Бейсике, но Лого отнесся к ней более строго. Я советую сделать наш оператор процедурой, аналогичной процедуре BOX, но с параметром. Это выглядит так:

```
TO POLYGON: SIDES
   REPEAT : SIDES [FD 40 RT 360/:SIDES]
END
```

Мы определили новый оператор POLYGON, при обращении к которому нужно указывать а р г у м е н т, т.е. значение параметра. Например:

```
POLYGON 6
```

рисует шестиугольник, а

```
POLYGON 8
```

– восьмиугольник. Обратите внимание, что чем больше сторон, тем большего размера многоугольник. Почему?

– Размер стороны у всех многоугольников один и тот же, он равен 40, поэтому периметр возрастает вместе с числом сторон. Пожалуй, 20-угольник уже не нарисуешь – он в экран не влезет.

– А можно нарисовать маленький 20-угольник?

– Да, если уменьшить длину стороны в процедуре POLYGON. Я бы поставил ее в зависимость от числа сторон: чем больше сторон, тем меньше длина каждой, чтобы периметр был примерно одинаков и многоугольник всегда умещался бы на экране.

– Грандиозно! Действуйте.

– Периметр приближенно равен длине окружности, диаметр которой – высота экрана, а именно 200, если измерять в точках. Умножаем на "пи" и округленно получаем 600. Следовательно,

```
TO POLYGON: SIDES
```

```
    REPEAT :SIDES [FD 600/:SIDES RT 360/:SIDES]
END
```

– Чтобы рисовать многоугольник в центре экрана, сместимся немного влево, но без рисования:

```
PENUP
LEFT 90
FORWARD 100
RIGHT 90
PENDOWN
```

– А теперь – 20-угольник:

```
POLYGON 20
```

Знаете, а ведь это почти окружность!

– Особенно если учитывать ограниченную точность нашего экрана.

– С увеличением числа сторон многоугольники почти не будут отличаться от окружности. Вот и решение задачи!

– Мы решили бы задачу быстрее, если бы вовремя вспомнили одно из определений окружности как "предела вписанных правильных многоугольников". Впрочем, наши упражнения, я думаю, не пропали зря.

Определение программистом процедур с параметрами фактически позволяет вводить собственные операторы с аргументами. В этом важное преимущество Лого перед Бейсиком. Использовать процедуры можно двояко: либо создавая новые операторы, т.е. новые *средства*, либо переходя от цели, конечной программы, к *подцелям*, ибо в процедуре вы можете упомянуть не определенную пока другую процедуру (вашу подцель). Это называется проектированием "снизу вверх" и "сверху вниз". Лучше всего использовать оба подхода, пока они не сомкнутся и вы не обнаружите, что ваша подцель достигнута – такой оператор уже есть.

Некоторые важные вопросы мы еще не обсудили. Вы можете заняться "языкознанием" и изучить другие языки программирования. Некоторые из них родственны между собой. Так, прямым прародителем Лого является язык Лисп, а Бейсик состоит в родстве с Фортраном. Вообще, мы пока не столкнулись с так называемыми *компиляторами*, т.е. трансляторами, которые не допускают непосредственного исполнения операторов, но зато имеют свои несомненные преимущества. Некоторую информацию о них вы почерпнете в предыдущем очерке.

Мир программ и программирования очень многообразен, и может статься, что вам придется с ним соприкоснуться. Если вы получили некоторое представление о нем и у вас появилось желание расширить ваши пока еще скромные познания, то цель этого очерка можно считать достигнутой.

ПРИКЛАДНЫЕ СИСТЕМЫ И РЕШЕНИЕ ЗАДАЧ

В.П. МАЗУРИК

Мы все пользуемся бытовыми средствами обработки информации – телефоном, телевизором, радиоприемником, и почти не прибегаем к помощи руководств и инструкций по работе с ними. Настольные калькуляторы – более сложные приборы, и все же при регулярном пользовании ими очень быстро исчезает необходимость в инструкции. Персональный компьютер – это новый прибор для обработки информации, который, как телефон или калькулятор, может быть для нас очень полезен и на работе и дома.

Часто бывает так, что более мощная машина проще в управлении. В этом нет никакого парадокса: часть мощности машины используется для упрощения управления. Персональный компьютер неизмеримо мощнее, богаче по своим возможностям, чем любой калькулятор. И во многих отношениях намного проще в использовании. Например, на персональном компьютере можно печатать тексты точно так же, как на обычной пишущей машинке. Расположение букв на клавиатуре компьютера стандартное, так что при наличии навыка можно печатать тексты "вслепую". Никакого дополнительного обучения не требуется. Компьютер как бы превращается в привычную пишущую машинку. Можно его превратить в калькулятор и произвести необходимые вычисления, после чего продолжить печатать текст, вставив в него полученные результаты вычислений. Можно превратить компьютер в "живой учебник", который начнет вас обучать приемам работы на нем самом, например расскажет (выдавая тексты на экран) о дополнительных возможностях по редактированию текстов или о том, как представить колонки чисел в таблице в виде красивого цветного графика на экране компьютера.

Все эти магические превращения компьютера осуществляются программным путем. В каждый момент времени компьютер исполняет некоторую программу. В зависимости от того, какая программа запущена, он и превращается то в пишущую машинку, то в калькулятор, то в построитель графиков. Программы такого рода, предназначенные для решения конкретного класса задач, называются прикладными программами и разрабатываются профессиональными программистами.

Рынок программного обеспечения для персональных компьютеров содержит сотни наименований прикладных программ различного назначения. Существуют программы для решения задач узкого класса, например для проектирования печатных плат ЭВМ, или для расчета оптимальной конструкции химического реактора, или для вычисления оптимального семейного бюджета на текущий месяц. Существует много программ универсального характера, таких, например, как текстовые редакторы, калькуляторы, электронные записные книжки и т.д.

Несмотря на сравнительно короткую историю развития программного обеспечения для персональных компьютеров уже возникли некоторые устойчивые классы программ, объединенные общей прикладной направленностью и единым интерфейсом с пользователем.

Одним из наиболее распространенных классов являются так называемые электронные таблицы. Они представляют собой своеобразные калькуляторы, сочетающие гибкие возможности по вычислениям и по манипулированию информацией: копированию, выборкам, упорядочению и т.д. Электронные таблицы должны стать одним из основных инструментов повседневной работы для банковских служащих, работников сферы управления и т.д.

Другим устойчивым классом являются программы для обработки текстов. Можно только удивляться тому, что за короткий срок их развития эти программы прошли путь от простых редакторов текста, известных узкому кругу профессиональных программистов, до текстовых процессоров массового использования, обладающих поразительным набором возможностей. Пожалуй, программы этого класса прогрессируют наиболее быстро, так как число их пользователей исчисляется сотнями тысяч.

Многим пользователям требуется более сложная обработка информации, чем простой набор текстов. Требуется, например, ввести большое количество однотипной информации (карточек служащих, чеков проданных товаров и т.д.) и затем провести ее упорядочение по определенным ключам, или выборку по какому-либо правилу. Все это обеспечивается программами ведения баз данных, которые составляют еще один устойчивый класс, обладающий своей спецификой, сложившимися традициями взаимодействия с пользователями, своими нерешенными проблемами.

Невозможно дать описание всего того богатства прикладных программ различного назначения, которые разработаны для персональных компьютеров. Выпускаются ежемесячные каталоги прикладных программ, содержащие сотни страниц текста и тысячи наименований программных продуктов.

Настоящий очерк можно считать кратким введением в использование персональных компьютеров для решения задач. Нашей целью является демонстрация наиболее распространенных программных средств решения задач на персональном компьютере, общего стиля его использования, а вовсе не перечисление каких-то конкретных классов задач и приемов их решения. Более того, главная идея, ради которой эти машины были изобретены, заключается в их названии: пользователь может выработать свои собственные, персональные приемы эффективного использования компьютера, ориентированные на специфику его сферы деятельности, личные склонности, уровень образования и желание повысить его и т.д.

Для демонстрации возможностей персональных компьютеров выбраны прикладные программы общего назначения, такие, как электронная таблица, редактор текстов, так называемая деловая графика и т.д. В качестве примера рассматриваются простые задачи из различных прикладных областей. Будет проведен анализ этих задач с точки зрения структур данных, которые возникают при их постановке и решении, а также специфики алгоритмов обработки этих данных. Конечной целью такого анализа будет выявление таких программных средств, реализация которых на персональном компьютере позволяет эффективно решать эти и аналогичные задачи.

Задача 1

Подготовка штатного расписания

В научно-исследовательский институт пришел проект новых должностных окладов для научных сотрудников. Перед руководством института возникает задача подготовки нового штатного расписания при условии, что общий фонд зарплаты института остается неизменным. Дирекция рассылает заведующим отделов записку, в которой указывается фонд зарплаты отдела и список новых должностных окладов вместе с правилами, регламентирующими назначение окладов сотрудникам. Указывается также срок, к которому требуется представить дирекции свои предложения. Заведующий отделом, получивший записку, начинает подготовку нового штатного расписания. Очевидно, что для этого ему нужны сведения по каждому сотруднику: должность, ученая степень, стаж и т.д. Все эти сведения можно запросить в виде справки стандартной формы из отдела кадров. Задача заведующего отделом состоит в том, чтобы перевести своих сотрудников на новые должности и назначить им оклады в пределах установленных вилок, причем общий фонд зарплаты не должен превышать известной величины.

Подготовленные предложения требуется оформить в виде записки заранее оговоренной формы, в которой будет содержаться

таблица новых окладов для сотрудников и, возможно, некоторые комментарии по поводу обоснованности предлагаемых назначений.

Предложения могут быть приняты дирекцией или отправлены на доработку. После того как все предложения будут готовы, должно быть составлено штатное расписание всего института. Этот процесс может потребовать внесения каких-то изменений на уровне штатных расписаний отделов, т.е. задача решается в несколько итераций, на каждой из которых проводятся одни и те же действия, но с разными исходными данными.

Проведем анализ действий заведующего отделом. Он получил список новых должностных окладов в виде некоторой таблицы. Новые правила таковы, что руководство вправе установить сотруднику должность и оклад в соответствии с этой таблицей. Ученая степень является необходимым условием для того, чтобы претендовать на высокую должность, но главным критерием при назначении на должность и выборе оклада в пределах разрешенной вилки являются профессиональные качества сотрудника.

Заведующий отделом решает проставить оценку каждому сотруднику в десятибалльной шкале. В соответствии с этой оценкой он назначает новые должности. Фонд месячной зарплаты отдела составляет 2000 руб. Возникает задача выбора окладов в пределах установленных вилок таким образом, чтобы не превысить месячный фонд.

Возможный вариант решения состоит в следующем. Каждому сотруднику оклад назначается по следующей формуле:

оклад = нижняя граница + вилка X оценка / 10

Другими словами, сотрудник получает оклад в пределах разрешенной для его должности вилки и тем больший, чем больше его профессиональная оценка.

В процессе вычислений заведующий отделом заполняет таблицу, в которой для каждого сотрудника указана его оценка, новая должность и оклад. Затем он вычисляет сумму окладов, и, если она не совпадает с выделенным месячным фондом зарплаты отдела, проводит корректировку окладов, повторяет все вычисления и составляет новую таблицу.

Остановимся в этой точке и проведем анализ действий заведующего отделом. Нетрудно видеть, что в процессе работы он пользовался простой формой структурирования данных – таблицей. Эта форма позволила в наглядном виде представить информацию, распределив ее по однотипным столбцам. В ячейках таблицы содержится информация трех различных видов. В первом столбце ячейки содержатся фамилии сотрудников, т.е. строки текста. Во втором столбце записаны числа – коэффициенты профессиональной оценки. Последний столбец представляет наибольший интерес. В нем тоже числа, однако нам известно, что они являются резуль-

татами вычислений по известной формуле, т.е. в действительности представляют функцию от числовых значений, записанных в этой таблице, а также в таблице новых должностных окладов.

Таким образом, и сама таблица, и проведенные вычисления очень просты. Таблицу можно нарисовать на листе бумаги, а вычисления провести с помощью карманного калькулятора. Однако вспомним о том, что полученные результаты должны быть пересчитаны и, вероятно, не один раз. Могут потребоваться еще какие-либо изменения в таблице. Покажем преимущество персонального компьютера в таких пересчетах.

Электронная таблица

Предположим, что персональный компьютер умеет работать с таблицами. Представим себе работу с такой электронной таблицей на компьютере. Мы подаем команду порождения таблицы, и на экране терминала появляется пустая таблица, ячейки которой пронумерованы: сверху – рядом букв (а, б, в, г и т.д.), а слева – рядом последовательных цифр, так что ячейка в левом верхнем углу получит имя а1, следующая за ней в той же строке – имя б1, далее – в1 и т.д. Ячейки второй строки получат имена а2, б2, в2 и т.д. Заполнение информации в таблице производится следующим образом. На экране виден курсор (т.е. прямоугольник, выделенный более ярким фоном, чем основное поле экрана), который находится в некоторой ячейке. Эта ячейка считается текущей. Можно гонять курсор по таблице, нажимая на клавиатуре клавиши "стрелка влево", "стрелка вправо", "стрелка вниз", "стрелка вверх". Расположив курсор в нужной ячейке, можно ввести в нее информацию, набирая текст на клавиатуре, например фамилию сотрудника. Затем нажимаем стрелку вправо, курсор оказывается в соседней справа ячейке, и мы вводим в нее число – оценку сотрудника. Таким образом, мы можем заполнить почти все ячейки и образовать в памяти компьютера таблицу (табл. 1).

В этой таблице ячейка в5 содержит число 6, ячейка г2 – слово "новая", а ячейки г15 и д15 содержат одинаковые строки символов вида "––––– –––", образуя таким образом завершающую линию под суммарными показателями.

Последний столбец таблицы пока остался пустым. Заполним ячейку д13 следующим образом: подведем к ней курсор и напишем: @**SUM** (д5. .д11). Эта запись означает, что содержимое ячейки является суммой значений всех ячеек в столбце над ней в диапазоне от д5 до д11. Символ "@" впереди означает, что мы вводим формулу. Если его не поставить, компьютер введет в ячейку символьную строку вида **SUM** (д5. .д11), а не формулу. Если теперь переместить курсор, например, в позицию д5 и ввести число 310,

Таблица 1

	а	б	в	г	д
1					
2		Ученая	Оценка	Новая	Новый
3		степень		должность	оклад
4					
5	Новиков А.Ф.	Кандидат	6	Старший	
6	Петров И.Т.	Доктор	8	Главный	
7	Федоров С.В.	Кандидат	4	Младший	
8	Суворов Ю.Д.		10	Старший	
9	Солдатов Н.К.	Кандидат	5	Старший	
10	Юрченко К.С.		6	Младший	
11	Григорьев В.П.	Кандидат	9	Ведущий	
12					
13				Сумма:	
14				Предел:	2000
15					

в ячейке д13 автоматически появится 310, так как такова в этот момент сумма всех чисел в столбце от д5 до д11. Перемещаем курсор на д6 и вводим число 440. Происходит мгновенное изменение значения в ячейке д13: оно становится равным 750. Уместен вопрос: куда же девалась сама формула, которую мы ввели в ячейку д13? Дело в том, что формула, как, впрочем, и любые другие значения, вводимые в ячейки, пишется в процессе ввода не в саму ячейку, а в специальной строке над таблицей. И только по завершении ввода введенное число или текст появляется в ячейке. В случае ввода формулы компьютер запоминает ее где-то у себя в памяти, а в ячейку вводит результат расчета по этой формуле. Поэтому при завершении ввода указанной выше формулы в ячейке д13 появится число 0.

Вспомним теперь о том, что в ячейки д5. .д11 мы хотим ввести не числа, а формулы. Трудность заключается в том, что в нашей таблице нет всех исходных данных. Часть из них хранится в таблице, где находится информация о новых должностных окладах.

Вернемся к моменту создания пустой таблицы в памяти компьютера. Возникает вопрос: сколько ячеек при этом создается? Будем считать, что количество ячеек вообще не ограничено. В действительности это, конечно, не так и реальное количество ячеек определяется размером памяти компьютера. Типичные размеры в существующих системах обработки таблиц таковы: две-три сотни по горизонтали и несколько тысяч по вертикали. Все это громадное пространство предоставляется пользователю и называется рабочим полем. На его территории можно располагать одновременно несколько таблиц. Именно это нам и требуется.

Таблица 2

	а	б	в	г
20				
21	Должность	От	До	Ограничение
22				
23	Главный научный сотрудник	400	450	Доктор наук
24	Ведущий научный сотрудник	300	400	Кандидат наук
25	Старший научный сотрудник	250	350	
26	Научный сотрудник	160	270	
27	Младший научный сотрудник	140	220	
28				
29				

Введем на рабочее поле таблицу новых должностных окладов, поместив ее, например, вслед за уже созданной таблицей, начиная с ячейки а21. Соответствующая часть рабочего поля приобретет вид табл. 2.

Теперь информации достаточно, чтобы написать формулы, определяющие значения в ячейках д5. .д11. В ячейку д5 вводим формулу вида + б25 + (в25 – б25) * в5 / 10. Знак плюс вначале ставится, чтобы отличить формулу от строки текста. Поскольку А.Ф. Новиков планируется на должность старшего научного сотрудника, мы выбираем нижнюю границу зарплаты из ячейки б25 и прибавляем к ней величину, пропорциональную его оценке (ячейке в5) в десятибалльной шкале. В момент завершения ввода формулы в ячейке д5 появляется число 310. Это же значение появится в ячейке д13. Аналогичные формулы будут записаны в остальные ячейки. В результате рабочее поле приобретет вид табл. 3.

Итак, первый этап работы завершен и первоначальный вариант назначения окладов сотрудникам составлен. Принципиальное отличие ситуации от той, когда вычисления проводились на бумаге, заключается в следующем.

Мы можем теперь экспериментировать с созданной электронной таблицей, производя какие-либо изменения на рабочем поле и немедленно получая результаты на экране персонального компьютера. Пусть, например, заведующий отделом решил ликвидировать превышение фонда зарплаты путем уменьшения зарплаты каждому сотруднику на величину, пропорциональную его зарплате в таблице, чтобы общее уменьшение составило 150 руб.

Нетрудно видеть, что если обозначить буквой х зарплату сотрудника в таблице, то нужно назначить новую зарплату в соответствии со следующей формулой: х * 2000 / 2150.

Заполним столбец е в таблице новыми значениями зарплаты. Вводим в ячейку е5 формулу следующего вида:

@INT (д5 * д14 / д13)

Таблица 3

	а	б	в	г	д
1					
2		Ученая	Оценка	Новая	Новый
3		степень		должность	оклад
4					
5	Новиков А.Ф.	Кандидат	6	Старший	310
6	Петров И.Т.	Доктор	8	Главный	440
7	Федоров С.В.	Кандидат	4	Младший	172
8	Суворов Ю.Д.		10	Старший	350
9	Солдатов Н.К.	Кандидат	5	Старший	300
10	Юрченко К.С.		6	Младший	188
11	Григорьев В.П.	Кандидат	9	Ведущий	390
12					
13				Сумма:	2150
14				Предел:	2000
15					
16					

Функция INT производит округление до целого числа. Аналогичные формулы введем в остальные ячейки столбца е. В ячейку е13 введем формулу суммы значений в столбце е (табл. 4).

Получен требуемый результат: суммарный месячный фонд зарплаты не превышает установленного предела. Предположим, что заведующий отделом решает повысить оклад К.С. Юрченко, так как планирует нагрузить его дополнительной работой. С этой целью он поднимает на две единицы его оценку в ячейке в10 и

Таблица 4

	в	г	д	е
1				
2	Оценка	Новая	Новый	
3		должность	оклад	
4				
5	6	Старший	310	288
6	8	Главный	440	409
7	4	Младший	172	160
8	10	Старший	350	325
9	5	Старший	300	279
10	6	Младший	188	174
11	9	Ведущий	390	362
12				
13		Сумма:	2150	1997
14		Предел:	2000	
15				

Таблица 5

	в	г	д	е
1				
2	Оценка	Новая	Новый оклад	
3		должность		.
4				
5	6	Старший	310	286
6	8	Главный	440	406
7	4	Младший	172	158
8	10	Старший	350	323
9	5	Старший	300	277
10	8	Младший	204	188
11	9	Ведущий	390	360
12				
13		Сумма:	2166	1998
14		Предел:	2000	
15				
16				

делает ее равной 8. На экране возникает новый вариант окладов с учетом всех ограничений. Например, автоматически гарантируется непревышение границ допустимой вилки, так как оценка остается меньше 10. Новые результаты показаны в табл. 5.

Заведующий отделом может удовлетвориться полученными результатами или продолжить эксперименты с таблицей. Поскольку все пересчеты производятся практически мгновенно, он может провести анализ большого количества вариантов за короткое время. Завершив эту работу, он составляет записку дирекции, в которой приводит полученные цифры и излагает в случае необходимости обоснование принятых решений. На этом этапе персональный компьютер тоже способен существенно облегчить и ускорить работу, но это мы обсудим ниже, когда речь пойдет об обработке текстовой информации.

Задача 2

Проектирование автомобиля

Предприятие занимается проектированием автомобиля. Принято решение об уменьшении общего веса автомобиля на 6%. Перед главным конструктором возникает задача выдать задание в отделы, которые занимаются проектированием отдельных узлов конструкции, на понижение веса этих узлов таким образом, чтобы обеспечить требуемый общий вес конструкции. Решение о проценте понижения веса каждого узла принимается им исходя из общего веса узла, оценки реальности понижения его веса на планируемую величину и многих других соображений. Понижение веса означает, вообще говоря, пропорциональное снижение

функциональных возможностей узла, так что главному конструктору приходится выбирать, какие возможности ему "дороже".

Приняв решение по отдельным узлам, главный конструктор сообщает полученные цифры в соответствующие конструкторские отделы. Начальник каждого отдела обязан обеспечить указанное снижение веса узла. При этом он фактически решает ту же задачу, но для тех компонент, из которых состоит проектируемая его отделом конструкция.

В большинстве случаев возникает потребность в корректировке этой задачи, так как выясняется невозможность обеспечить указанное снижение веса какого-либо узла, или, наоборот, удается путем применения оригинального конструктивного решения дать дополнительное снижение веса узла и "добавить" разрешенный вес для более приоритетного узла.

Несмотря на внешнее различие, эта задача по своей сути очень близка к задаче 1, и для ее решения снова может быть применена электронная таблица. В самом деле, главный конструктор, получивший задание снизить общий вес проектируемого автомобиля на 6%, поступает следующим образом. Он составляет таблицу отдельных узлов конструкции, в которую входят такие позиции, как кузов, двигатель, коробка передач и т.д. С каждым узлом связан его вес и некий показатель или система показателей, задающих особенности конструкции узла, возможность его модификаций и оценку их трудоемкости, степень завершенности проекта узла и т.д. Все эти данные он заносит в таблицу, которая становится основным инструментом дальнейшего анализа. Целью анализа является разработка заданий на снижение веса каждого узла либо в абсолютных показателях, либо в процентах от существующего веса. Действия с таблицей в процессе анализа примерно те же, что и в первой задаче: изменение значений в отдельных ячейках и анализ влияния этих изменений на другие значения, т.е. моделирование различных вариантов распределения веса по узлам. Эффект применения электронной таблицы заключается в существенном снижении времени на оценку вариантов по сравнению с их ручным расчетом, а это приводит в конечном счете к улучшению качества конструкции, так как можно за короткое время провести анализ большего количества вариантов.

Электронная таблица есть средство организации данных, придания им определенной структуры, упрощающей их анализ и обработку. Существенно то, что средства таблицы позволяют менять структуру, выбирая наиболее подходящий вариант для решения конкретного вопроса. Например, при описании конструкции автомобиля можно было ввести в таблицу информацию о распределении веса не только по конструктивным узлам, но также и по функциональным подсистемам, таким как тормозная систе-

ма, электросистема и т.д. При этом с помощью таблицы легко получить ответы на вопросы типа: "какая часть веса электросистемы приходится на двигательную установку?" или "сколько процентов общего веса автомобиля составляет вес тормозной системы?". Все расчеты такого рода можно ввести в таблицу в виде формульных зависимостей точно так же, как это было показано для задачи 1. Нетрудно понять, насколько эти средства упрощают анализ вариантов по снижению общего веса и делают его более гибким и целенаправленным.

Удачный выбор структуры данных может существенно облегчить восприятие этих данных человеком, работающим за терминалом персонального компьютера. Однако возможности персональных компьютеров по представлению данных не ограничиваются только средствами удобного распределения колонок чисел по экрану. Во многих случаях оказывается удобным перевести информацию из числовой в образную, графическую форму.

Деловая графика

Предположим, что главный конструктор ввел информацию в электронную таблицу

Таблица 6

	а	б	в	г	д	е
1		Вес кон-	Электро-	Тормозная	Топливная	Вес
2		струкции	система	система	система	узла
3						
4	Кузов	173.4	24.6	12.7	48.4	259.1
5	Двигатель	278.6	41.4		25.2	345.2
6	Трансмиссия	136.3	7.2			143.5
7	Передн. подвеска	116.8		28.3		145.1
8	Задняя подвеска	152.7		42.6		195.3
9						
10	Вес подсистемы:	857.8	73.2	83.6	73.6	
11						
12				Общий вес:		1088.2
13				Требуется снизить на:		65.3
14						

Представим ту же информацию в более наглядной форме. На рис. 1 изображена круговая диаграмма, представляющая вклад в процентном отношении отдельных узлов в общий вес конструкции. Конструктор видит, что наибольший вклад дает двигатель (31.7% общего веса). Но менять конструкцию двигателя очень сложно. Второй по величине вклад дает кузов. Это главный пре-

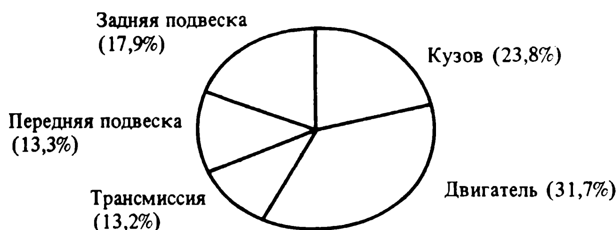


Рис. 1. Распределение веса по узлам



Рис. 2. Распределение веса по функциональным системам в узле "кузов"

тендент на снижение веса. Видно также, что тяжела задняя подвеска, ее тоже можно облегчить. Проводим анализ вклада весов функциональных подсистем в общий вес кузова. Получаем диаграмму, изображенную на рис. 2. Становится понятно, что надо в первую очередь уменьшать вес конструкции кузова. Можно обратить внимание также на топливную систему.

Понятно, что никакой новой информации при выводе этих изображений не возникает. Удобство заключается просто в смене точки зрения на информацию. Преимущество графической формы представления проявляется особенно ярко тогда, когда числовой информации много, и в то же время важны не столько сами числа, сколько общая картина. Сравните, например, картинку синусоиды и таблицу, в которой она же описана в виде длинного ряда пар чисел. По картинке сразу видно, что имеется некий колебательный процесс, причем видна и его амплитуда и частота.

Возможность выдачи информации в графическом виде реализуется прикладными программами так называемой деловой графики. Деловая графика тесно связана с электронной таблицей. Она позволяет мгновенно менять точку зрения с числовой на графическую и обратно. Важным моментом является факт интеграции этих средств в единое целое. При описании графической формы выдачи информации на экран мы ссылаемся на данные в электронной таблице. Эти ссылки запоминаются, и далее

8—92

изменение каких-либо числовых значений в таблице приводит к автоматическому изменению соответствующего графического образа на экране. Это позволяет в процессе анализа вариантов просматривать их в наиболее наглядной графической форме.

Рассмотрим процесс порождения графического образа на экране персонального компьютера, соответствующего круговой диаграмме на рис. 1.

Работа с таблицей производится следующим образом. Нажимаем на клавиатуре компьютера клавишу с названием "меню". На экране в строке, расположенной выше основного поля таблицы, возникает меню, содержащее следующие позиции: "копировать двигать удалить вставить график формат диапазон".

Позиции меню соответствуют действиям, которые можно произвести над таблицей. Одно из них, обозначенное ключевым словом "график", соответствует построению графических образов. Курсор расположен в первом поле меню. Над строкой меню расположена строка информации: "создание копии ячеек таблицы". В ней дается объяснение смысла той позиции, в которой в данный момент находится курсор. Двигаем курсор вправо, нажимая на клавиатуре клавишу "стрелка вправо". По ходу движения текст в строке информации меняется. Добираемся до позиции "график". В строке информации появляется текст "построение графиков". Нажимаем на клавишу "исполнение команды". На месте старой возникает новая строка меню и выше нее – строка информации, объясняющая первую позицию меню:

выбор типа графика

тип диапазон формат надписи заголовок демонстрация

Остальные позиции меню имеют следующий смысл:

указание наборов данных таблицы для построения графика;

выбор типа линий и узловых точек на графике;

разметка позиций графика надписями;

текст, помещаемый над графиком;

демонстрация построенного графика.

Выбираем позицию "тип". Возникает новое меню, в котором перечисляются возможные типы графиков, известные электронной таблице:

линейный гистограмма-1 гистограмма-2 X–Y диаграмма

Эти позиции меню имеют следующий смысл:

линейный график (значения по оси X могут быть нечисловыми, график изображается точками и соединяющими их линиями);

обычная гистограмма (столбцы располагаются рядом);

стековая гистограмма (столбцы – один над другим);

график изображается на осях координат X–Y (обе оси – числовые);

круговая диаграмма.

Выбираем вариант "диаграмма". Теперь надо вернуться к меню предыдущего уровня. Это делается нажатием клавиши "возврат". Выбираем позицию "диапазон". Возникает меню следующего вида:

X фун-1 фун-2 фун-3 фун-4 фун-5 фун-6

Это меню предлагает определить наборы чисел в таблице, которые будут играть роль значений независимой переменной (она условно обозначается буквой X) и функций. На одном графике можно построить одновременно несколько различных функций от переменной X.

Выбираем позицию X в меню. Курсор перепрыгивает со строки меню на поле ячеек электронной таблицы. Таблица приглашает нас отметить курсором тот диапазон данных в какой-либо строке или столбце таблицы, который должен играть роль значений независимой переменной. Подводим курсор к ячейке a4 (см. табл. 6), затем нажимаем клавишу "точка" (отмечая начало диапазона) и двигаем курсор вниз до позиции a8. Ячейки a4 . . . a8 по мере движения по ним курсора последовательно выделяются ярким фоном. Нажимаем клавишу "исполнение команды". Электронная таблица запоминает указанный диапазон и возвращает курсор в меню (яркий фон в таблице при этом пропадает).

Теперь нужно задать числовые значения функции, соответствующие каждому выбранному значению независимой переменной. Выбираем позицию "фун-1" в меню и отмечаем в таблице ячейки e4...e8. Функция задана. В этот момент электронной таблице известно все, что нужно для построения графика: его тип и данные, задающие функцию. Нажимаем клавишу "возврат" и попадаем в меню предыдущего уровня. Выбираем команду "демонстрация". На экране компьютера гаснет таблица и мгновенно возникает картинка, на которой изображена подготовленная нами круговая диаграмма (см. рис. 2).

Предположим, что мы решили изобразить ту же функцию в виде линейного графика. Нажимаем "возврат" и попадаем в меню, содержащее позицию "тип". Выбираем эту позицию и оказываемся в меню, содержащем типы графиков. Выбираем позицию "линейный" и возвращаемся в предыдущее меню. Выбираем команду "демонстрация". На экране возникает график, изображенный на рис. 3. Заметим, что электронная таблица позаботилась об оформлении графика: слева по координате Y показана шкала значений, выбран наиболее подходящий масштаб выдачи, по оси X расставлены названия узлов.

Построим с помощью электронной таблицы "интегральный образ" распределения веса по функциональным подсистемам.

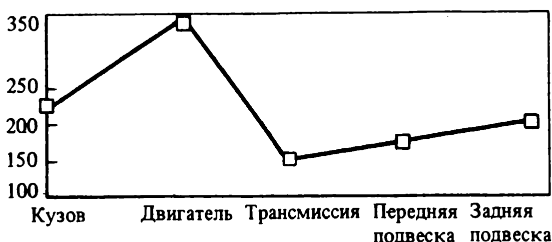


Рис. 3. Линейный график



Рис. 4. Гистограмма

Выбираем тип графика "гистограмма-1" и задаем следующие диапазоны значений:

X: ячейки а4...а8
фун-1: ячейки б4...б8
фун-2: ячейки в4...в8
фун-3: ячейки г4...г8
фун-4: ячейки д4...д8

Далее задаем надписи при каждой из функций (команда "надписи" в меню) и заголовок "Распределение веса функциональных подсистем по узлам" (команда "заголовок" того же меню). В результате при выполнении команды "демонстрация" получаем график, изображенный на рис. 4.

Возможности деловой графики преследуют единую цель: улучшить восприятие информации человеком, сделать ее более наглядной и выразительной. Например, если на персональном компьютере установлен цветной дисплей, можно все картинки вы-

давать в цвете, причем цвета вы выбираете сами. Картинки можно запоминать в памяти компьютера и затем в нужный момент вызывать на экран, или сделать их копию на листе бумаги, воспользовавшись принтером.

Основной целью деловой графики, как уже было сказано выше, является оперативная выдача информации в графической форме в процессе анализа задачи при ее решении средствами электронной таблицы. Такая постановка задачи не предьявляет слишком высоких требований к качеству выдаваемых на экран картинок. Главным критерием здесь является быстрота подготовки и отображения графических образов, соответствующих оперативно изменяющейся числовой информации.

Существует большое число прикладных программ, предназначенных для графических приложений другого рода. Например, существуют пакеты программ для проектирования узлов машиностроительных конструкций. Эти пакеты обеспечивают построение чертежей высокого качества либо на экране компьютера, либо на графопостроителе. В других приложениях, где важен цвет, используются другие пакеты, обеспечивающие выдачу на цветной экран персонального компьютера различных схем, карт и т.д. В лучших образцах персональных компьютеров количество оттенков цветов достигает нескольких тысяч. Однако для широко распространенных компьютеров эта цифра обычно не превышает двух десятков.

Задача 3

Учет лекарственных средств

В медицинском учреждении существует сложившаяся форма представления информации о лекарственных средствах. Форма эта достаточно сложна. Описание лекарственного средства содержит такие пункты, как его общая характеристика, фармодинамика, фармакокинетика, показания к применению, режим дозирования, побочные эффекты, противопоказания и т.д. Каждый пункт состоит из множества подпунктов. Требуется разработать справочную систему для быстрого доступа к информации о лекарственных средствах. Система должна обеспечивать возможность пополнения и корректировки информации в своем архиве, осуществлять поиск записей о лекарственных средствах, обладающих указанными свойствами, производить упорядочивание записей, например в алфавитном порядке по наименованиям лекарственных средств, и т.д. Требуется также, чтобы информация выводилась на терминал в удобной для чтения форме. Например, неудачным решением является вывод на экран терминала сразу всей информации о лекарственном средстве, так как

вся она одновременно не нужна, и избыток информации только затрудняет ее восприятие. Гораздо логичнее вывести перечень упомянутых выше пунктов и затем показать информацию по одному из них по выбору пользователя.

Характерной чертой этой задачи является необходимость хранить и обрабатывать большое количество однородной информации – по одной записи стандартного формата на каждое лекарственное средство. Под обработкой понимаются операции внесения новых записей или корректировки уже введенных, поиск записей, удовлетворяющих заданным критериям, упорядочение записей и т.д. Для персональных компьютеров разработано большое количество прикладных программ, предназначенных для решения такого рода задач. Это так называемые программы или системы управления базами данных.

Базы данных

Средства обработки информации играют ведущую роль в программном обеспечении персональных компьютеров. Особенно интересны в этом отношении программы управления базами данных. Эти программы позволяют качественно ускорить и упростить процедуры обработки информации. С их помощью можно эффективно производить такие виды обработки данных, которые были просто немыслимы при традиционной ручной обработке. Появление нового инструмента всегда приводит к формулировке новых задач, к осознанию новых возможностей и ресурсов, скрытых в изучаемой области. Это в полной мере относится и к базам данных. Эта ветвь программного обеспечения персональных компьютеров быстро развивается и оказывает большое влияние на все другие его компоненты и на общее понимание перспектив развития безбумажной информационной технологии.

Покажем, как использовать базу данных для решения задачи учета лекарственных средств. Пользователь вызывает программу управления базой данных и начинает с ней диалог. Вначале он задает определение формата записи, т.е. описание полей, из которых состоит запись. В это описание входят имена полей, вид информации, заносимой в поле (число, текст), размер каждого поля и т.д. Можно задавать ограничения на значения полей, и тогда происходит автоматический контроль вводимых значений по этим ограничениям. Система просто не даст ввести неправильное значение. Например, в группе полей, определяющих режим дозирования лекарства, можно ввести поля "разовая доза" и "суточная доза" и предусмотреть, чтобы величины в этих полях не превосходили некоторых значений.

После того как формат записи определен, можно начать про-

цедуру ввода информации о лекарственных средствах в компьютер. На экране возникает пустая форма, состоящая из незаполненных полей записи. Каждое поле помечено своим именем. Курсор помещается в первом поле, и программа ждет ввода значения с клавиатуры компьютера. Когда значение введено и оно оказалось допустимым для этого поля (работает автоматический контроль), курсор перемещается на следующее поле записи и т.д. Расположение полей на экране и порядок их обхода определяет сам пользователь в процессе описания формата записи. Совсем не обязательно вводить значения всех полей. Есть также возможность определять поля, значения которых устанавливаются заранее самой программой (но, конечно, по заказу пользователя). При возникновении на экране пустой формы эти поля уже заполнены. Естественно, их значения, как, впрочем, и значения любых других полей, можно переопределить в процессе заполнения формы.

Закончив заполнение формы, можно пополнить полученной записью базу данных. Это делается нажатием клавиши "ввести запись". При этом на экране опять возникает пустая форма, и можно начать ввод следующей записи и т.д.

Созданная база данных является не только хранилищем информации. Она позволяет с этой информацией активно работать: делать выборку подмножества записей, удовлетворяющих некоторому критерию, упорядочивать записи, добавлять, уничтожать или корректировать записи в базе и т.д. Например, можно дать команду выборки всех лекарственных средств, имеющих определенное показание к применению. Можно усложнить условие, добавив требование, чтобы при поиске исключались лекарства с указанным побочным эффектом. Можно расположить выбранные записи в алфавитном порядке по названиям лекарственных средств.

Записи могут содержать большое число полей. Далеко не всегда требуется одновременное появление всех полей на экране при просмотре записей. Известно, что избыток информации затрудняет ее восприятие. В базах данных существуют средства настройки формата выдачи данных при просмотре записей. Можно заранее подготовить себе несколько таких форматов для разных случаев и вызывать их по мере необходимости. Каждый формат как бы меняет точку зрения на записи, выделяя требуемый набор полей и располагая их на экране определенным образом. Например, можно подготовить формат, в котором сосредоточены только поля, задающие общую характеристику лекарственного средства: его название, химическую формулу, физико-химические характеристики, такие, как растворимость, гидрофильность и т.д.

Прикладные программы для ведения баз данных обычно об-

ладают развитыми средствами организации диалога с пользователем. Взаимодействие с ними выглядит примерно так же, как в случае электронных таблиц: на экране располагается меню, из которого движением курсора можно выбрать подходящую команду и исполнить ее или запросить у системы помощь в виде информации о ситуации или команде и т.д.

Связь с электронными таблицами не ограничивается только внешним сходством организации взаимодействия с ними. Нетрудно заметить, что между базами данных, основанными на таких понятиях, как запись, поле и т.д., и электронными таблицами с их ячейками, строками и столбцами существует глубокое структурное сходство. Это сходство эффективно используется в интегрированных системах, включающих в себя электронную таблицу и базу данных. Последовательные ячейки в строке электронной таблицы можно считать полями записи в базе данных. При этом вся база данных превращается в набор строк таблицы. Такое превращение открывает возможность использовать одновременно все средства, предоставляемые и электронной таблицей, и программой ведения базы данных. Электронная таблица позволит нам мгновенно пересчитать значения полей типа знаменитого бухгалтерского "итого", одной командой удалить поле из всех записей (это сводится к удалению столбца в таблице) или средствами деловой графики представить какие-то данные в наглядной форме. В свою очередь, средствами базы данных можно сделать поиск записей по критерию или их упорядочение. База данных превращается в своеобразную "призму" над информацией, упрятанной в электронную таблицу. Сквозь эту призму мы можем рассматривать дополнительные аспекты информации. Такая смена точки зрения зависит от целей, которые мы в данный момент перед собой ставим, и может производиться многократно в процессе анализа и решения задачи.

В процессе анализа предыдущих задач был сделан вывод о необходимости интеграции электронной таблицы и программ деловой графики. Теперь к этому списку можно добавить программы ведения баз данных.

Вернемся к анализу двух первых задач. В этих задачах окончательная обработка информации предполагает подготовку отчетов, содержащих полученные числовые результаты и некоторый поясняющий текст. Возникает два вопроса: каким образом средствами персонального компьютера подготовить текст отчета и как включить в этот текст полученные числовые результаты. Конечно, можно просто перенести эти результаты в текст вручную, набирая их на клавиатуре компьютера точно так же, как и остальной текст. Если числовой информации немного, это не займет много времени. Если же подготовленные для отчета таблицы

велики, такой вариант совершенно не годится, так как он обесценивает те преимущества в скорости, которые дает электронная таблица. Очевидно, требуются средства интеграции, которые дали бы возможность одновременного использования и электронной таблицы, и редактора текстов. Это очередной случай, когда мы приходим к идее интеграции прикладных программ, имеющих различную цель.

Преимущества от объединения средств деловой графики, электронной таблицы и базы данных уже обсуждались выше. Если теперь добавить к ним текстовый редактор, то получится очень эффективный комплексный инструмент для решения задач на компьютере, охватывающий все стадии решения. Идея интеграции получила свое программное воплощение в уже упоминавшихся интегрированных системах, которые являются одним из наиболее популярных классов прикладных программ среди пользователей персональных компьютеров. Ниже будет дано описание таких систем. Будет, в частности, показано, как с их помощью завершить решение описанных задач.

Остановимся теперь более подробно на средствах обработки текстов, предоставляемых пользователю персональным компьютером.

Текстовый редактор

Персональный компьютер является очень удобным средством для обработки произвольной текстовой информации. Вся книга, которую читатель держит в руках, была написана на персональном компьютере. Автор этих строк не совсем корректно употребляет прошедшее время, так как в данный момент он именно тем и занимается, что вводит их в память компьютера. Происходит это совершенно так же, как при работе с обыкновенной пишущей машинкой. На клавиатуре, которая по расположению клавиш в точности повторяет пишущую машинку, нажимаются буквы, и они появляются на экране. Разница начинает ощущаться в тот момент, когда сделана ошибка.

Совершенно справедливо сказано: "что написано пером, не вырубишь топором". Владельцам персональных компьютеров повезло: компьютер пишет не пером и не на бумаге. На экране в момент набора текста виден мигающий курсор, который отмечает текущую позицию. В этой позиции появится следующая нажатая на клавиатуре буква. Но можно нажать не букву, а клавишу "стрелка вверх", и курсор переместится вверх, на предыдущую строку. Двигая курсор стрелками, подводим его к тому месту в слове, где сделана ошибка, и вставляем пропущенную букву. При этом автоматически весь хвост строки справа сдвинется на одну позицию. Можно теперь вернуться назад и продолжить печать текста.

Если при наборе очередной строки не остается места на экране, весь текст автоматически продвинется на одну строку вверх. Первая строка пропадет (с экрана, но не из памяти компьютера!), и внизу образуется пустая строка для продолжения работы.

Итак, первая возможность прикладных программ обработки текстовой информации (их обычно называют текстовыми процессорами, или редакторами) заключается в наличии средств произвольных перемещений по тексту и внесения любых исправлений. Двигаться можно медленно с помощью клавиш-стрелок, или быстро. Второй случай особенно удобен, если набирается большой текст, который целиком не помещается на экране (на нем помещается примерно 20 строк), и требуется, например, переместиться в начало текста. В любом редакторе обязательно есть средства для таких быстрых перемещений. Например, в том редакторе, который автор использует в данный момент, для перемещения в начало текста достаточно нажать на клавишу "бесконечное повторение" и затем на клавишу "стрелка вверх". Редактор "поймет", что требуется переместить курсор как можно выше вверх, т.е. на первую строку текста, выполнит это перемещение и мгновенно выведет на экран строки текста, начиная с первой.

Можно такого рода перемещения представлять себе следующим образом. Есть очень длинный текст, который строка за строкой записан в памяти компьютера, как на рулоне бумаги. Экран – это окно размером в 20 строк. Окно можно перемещать по тексту и произвольным образом менять текст, видимый в пределах окна. Интересной возможностью текстовых редакторов является организация работы с редактируемым текстом с использованием одновременно нескольких окон, которые настраиваются на различные фрагменты текста. Окна могут иметь различный размер. На экране может помещаться только одно окно или несколько окон одновременно. Последний случай очень удобен, если требуется сравнивать два фрагмента текста, расположенных далеко друг от друга.

При наборе текста очередной строки мы рано или поздно упираемся в правое поле, ограничивающее окно. На пишущей машинке в этот момент надо передвинуть каретку на начало строки. Понятно, что "умный" редактор с такой простой задачей справляется сам и переводит курсор на новую строку. Если при этом мы оказались на середине слова, то оно целиком "перепрыгивает" на следующую строку. Можно заранее договориться с редактором, чтобы он в таких случаях выравнивал строку по правому полю за счет увеличения пробелов между словами. Но учитывая возможные исправления по тексту, например вставку пропущенных слов или букв, можно этого и не делать, а выполнить выравнивания одним приказом, после того как весь текст готов и проверен.

Очень полезными оказываются команды редактора, которые позволяют работать не с отдельными символами, а с большими блоками текста. Блоки определяются по-разному в разных редакторах. Блоком может быть, например, совокупность подряд идущих строк. Блок можно выделить (отметить) по тексту экрана, удалить, переместить, скопировать в нужную позицию.

Во всех редакторах существуют команды так называемого контекстного поиска и редактирования. Предположим, что принято решение изменить терминологию в книге и вместо слов "персональный компьютер" везде писать "персональная ЭВМ". Это можно сделать следующим образом. Перемещаемся на начало текста и даем редактору приказ:

НАЙТИ: персональный компьютер
ЗАМЕНИТЬ НА: персональная ЭВМ

Приказ дается нажатием клавиши "контекстный поиск". Над полем окна редактора, в котором помещается редактируемый текст, возникают ключевые слова приказа: НАЙТИ: и ЗАМЕНИТЬ НА:. За каждым словом отмечена позиция для набора текстов: что искать и на что менять. Курсор оказывается в первой позиции. Набираем слова "персональный компьютер". Перемещаем курсор во вторую позицию и набираем слова "персональная ЭВМ". В этот момент можно начать исполнение приказа, нажав клавишу "исполнить команду". По всему тексту произойдет изменение первой фразы на вторую.

Можно учесть наличие падежей и искать контекст "персональн", не задавая вообще фразы ЗАМЕНИТЬ НА. Редактор будет последовательно подводить курсор к каждому следующему вхождению в текст указанного фрагмента при очередном нажатии клавиши "повторение". Дальше можно производить замену вручную, учитывая падеж, множественное число и т.д.

Хорошие редакторы позволяют писать вперемешку слова русским и латинским шрифтом. Это особенно удобно при работе над научными статьями и книгами. Некоторые редакторы позволяют делать различные разметки текста: выделять слова курсивом, подчеркивать слова или фразы, реверсировать изображение (белым по черному) и т.д. Для математиков созданы редакторы, которые умеют изображать на экране специальные математические значки вроде знака интеграла, верхних и нижних индексов переменных и т.д.

Существуют возможности по форматированию готового текста для его выдачи на бумагу. Можно сообщить редактору размер полей слева и справа, количество строк на странице, интервал между строками, нужно ли нумеровать страницы, нужно ли производить выравнивание текста по правому полю и т.д.

Готовый текст можно записать во внешнюю память, например, на гибкий диск. Можно собрать текст из кусков, записанных в различные файлы, или, наоборот, разбить большой текст на куски, чтобы легче было их редактировать.

Программы текстовых редакторов чрезвычайно интенсивно развиваются. Интерес к ним велик потому, что широк рынок сбыта. Трудно вообразить себе современного человека, который чего-нибудь не пишет – книгу, статью, письмо. Текстовые редакторы нужны всем. По-видимому, они составляют крупный камень в фундаменте новой информационной технологии, которая должна прийти с широким распространением персональных компьютеров.

В процессе анализа перечисленных выше задач неоднократно отмечалась потребность в интеграции программных средств различного назначения.

Интегрированные системы

Одним из наиболее интересных классов прикладных программ для персональных компьютеров являются так называемые интегрированные системы.

Не существует точного определения интегрированной системы. Причина этого заключается, по-видимому, в том, что невозможно однозначно сформулировать цели интеграции. Наименее точное и наиболее близкое к истине определение можно сформулировать так: "Интегрированной называется система, которую пользователь включает утром и работает на ней весь день, не испытывая никакой потребности в других программах". Это определение дал Митч Кейпор, президент одной из наиболее известных западных фирм по разработке программного обеспечения для персональных компьютеров – фирмы Лотус Девелопмент.

Можно выделить три типа систем такого рода: семейства пакетов, интегрированные системы и операционные оболочки. Их основное отличие друг от друга заключается в степени интеграции.

Первые попытки объединения прикладных программ в единую систему были сосредоточены на интеграции по информации. Отдельные компоненты (прикладные пакеты) системы могли обмениваться файлами, что обеспечивалось либо разработкой единых стандартов на структуру файлов, либо включением в систему специальных программ трансляции файлов из одного формата в другой.

Следующий этап интеграции предусматривал более тесную связь отдельных компонент. Понятно, что единый формат файлов является необходимым, но далеко не достаточным условием

интеграции компонент. Нужно, чтобы одна программа системы "понимала" информацию, передаваемую ей другой программой. Например, программа графического отображения информации должна уметь находить исходные данные в файле, подготовленном программой ведения электронной таблицы. Этот подход реализован в семействах пакетов. Семейство пакетов разрабатывается обычно одной фирмой. Вопросы объединения пакетов при этом решаются на стадии их разработки, а не путем навешивания логических заплат, как это часто бывает при стыковке автономно созданных пакетов.

Интегрированные системы являются следующим логическим шагом в развитии интеграции программных средств. Такие системы объединяют в рамках одной программы несколько прикладных пакетов. Существует общепринятый набор средств, которые включаются в современные интегрированные системы. Этот набор обычно включает пять компонент: пакет текстовой обработки, пакет работы с электронной таблицей, графический пакет, систему управления базой данных и пакет телекоммуникации.

Преимущества интегрированных систем заключаются в первую очередь в том, что они удобны для пользователя. Система имеет единый интерфейс с пользователем, которому не надо учить различные наборы команд, относящиеся к различным пакетам. Например, в интегрированной системе существует единая команда перемещения фрагментов текста, которая работает как при редактировании текстов, так и при использовании электронной таблицы. Интегрированные системы удобнее, чем разделенные пакеты, потому что пропадают все проблемы, связанные со стыковкой пакетов. Это, в частности, увеличивает скорость работы, так как не требуются долгие перекачки файлов через внешнюю память. Не тратится время и на переключение с одного пакета на другой – все они в интегрированных системах одновременно помещаются в оперативной памяти.

Последнее обстоятельство открывает также и список недостатков интегрированных систем. Первый недостаток – требования к оперативной памяти. Как правило, для работы с такими системами требуется не менее 250–300 Кбайт памяти. Другой недостаток, менее существенный, заключается в том, что в интегрированных системах каждый пакет работает чуть хуже, чем лучшие образцы в своем классе. Эта очевидная плата за интегрированность далеко перекрывается преимуществами, из нее же вытекающими. Более существенным является то, что эти системы "закрыты". В них трудно внести изменения или дополнения с целью подстроиться под требования пользователя.

Макросредства, которые включаются в современные интегри-

рованные системы для решения этой проблемы, решают ее только частично. Проблема модификаций наиболее остро стоит перед разработчиками проблемно-ориентированных систем. Их цель – включить в свои программы все те удивительно богатые средства, которые сосредоточены в современных интегрированных системах, "наложить" эти средства на прикладную специфику своих программ.

Эту проблему легко решают интегрированные системы третьего типа – операционные оболочки, которые работают по принципу "сделай сам". Пользователь сам решает, какой набор пакетов его устраивает, и собирает эти пакеты на своем компьютере. Операционная оболочка является программной прослойкой между прикладными пакетами и операционной системой. Ее можно считать расширением операционной системы, имеющим целью поддержку процесса интеграции прикладных пакетов. Использование операционных оболочек требует определенной квалификации от пользователя. Если для работы с прикладными пакетами и интегрированными системами требуются абсолютно минимальные знания возможностей операционной системы, то для операционных оболочек это не так. Но они и предназначены для тех сравнительно редких пользователей, которых не устраивает существующий набор прикладных систем для персональных компьютеров. Ниже будет дано краткое описание нескольких популярных интегрированных систем и операционных оболочек.

Вернемся к примерам. В задаче 1 заведующий отделом сначала делает расчет нового штатного расписания средствами электронной таблицы, а затем начинает подготовку отчета для дирекции.

Предположим, что он работает с интегрированной системой, включающей в себя электронную таблицу и редактор текстов. Идея интеграции реализуется не просто объединением программ "в одну кучу". В распоряжение пользователя предоставляется единое рабочее поле. Есть возможность менять точку зрения на это рабочее поле, превращая его попеременно то в электронную таблицу, то в поле редактора текстов. Объявляем его сначала электронной таблицей и проводим расчеты вариантов штатного расписания. Выше было показано, как это делается. Затем переходим в режим редактора текстов. Разметка рабочего поля при этом меняется: пропадает нумерация ячеек электронной таблицы и курсор уменьшается до размеров одного символа. Важно подчеркнуть, что вся информация, созданная в режиме электронной таблицы, остается на рабочем поле. Выбираем на рабочем поле любое свободное место и пишем на нем текст отчета, пользуясь всеми средствами редактора текстов.

Возвращаемся в режим электронной таблицы. На экране опять возникает нумерация ячеек. Теперь таблица содержит результаты

расчетов и введенный текст. Вызываем меню и выбираем команду "двигать", задавая ей в качестве аргументов диапазон ячеек, в которых расположены результаты расчетов (что двигать), и позицию внутри текста отчета (куда двигать). Электронная таблица выполняет эту команду, и отчет готов. Осталось выдать его на печать. Соответствующая команда есть в меню.

Последовательная реализация идеи единого рабочего поля в интегрированных системах потребовала разработки целого ряда новых интересных возможностей. Одна из них – это организация многооконного взаимодействия с рабочим полем. Экран персонального компьютера не очень велик и в большинстве случаев не вмещает одновременно всех таблиц, которые мы создаем на рабочем поле. Как в таком случае следить за изменением значений одних полей при изменении значений в других полях? Для этого и существуют окна.

Если мы занимаемся анализом последствий изменения значений в какой-то группе позиций таблицы на значения столбца суммарных показателей, создадим два окна. Первое настроим на те позиции, которые собираемся варьировать, а второе – на указанный столбец. Поместим оба окна одновременно на экране. Теперь, несмотря на то, что эти две области рабочего поля расположены далеко одна от другой, мы видим их одновременно. Если теперь вспомнить, что в нашем распоряжении есть средства деловой графики, то можно породить третье, графическое окно, и выдавать в него, например, гистограмму суммарных показателей как функций от варьируемых ячеек таблицы. Из этого примера видно, что в интегрированных системах нужно ввести типизацию окон, т.е. уметь настраивать окна на тип выдаваемой в них информации. Нам уже известны четыре различных типа, которые было бы удобно приписывать окнам: тип электронной таблицы, тип текстового поля, графический тип и, наконец, тип окна для работы с записями базы данных. Все эти типы окон реализованы в интегрированных системах. В качестве примера на рис. 5 изображен вид экрана в тот момент, когда одновременно происходит работа с тремя различными окнами: окном электронной таблицы, текстовым окном и графическим окном.

В меню интегрированных систем есть команды для работы с файлами. В конце рабочего дня все полученные результаты можно записать в файл, чтобы на следующий день продолжить работу с прерванного места. Файлы помещаются на устройствах долговременного хранения информации, например на гибких дисках. Если работа завершена, то ее результаты можно тоже записать на дискету и передать ее в соседний отдел, где эти результаты ждут. В файлах можно хранить любую информацию: тексты, таблицы и даже графические образы. Можно, например, породить на рабо-



Рис. 5. Вид экрана с тремя окнами

чем поле графическое окно и выдать в него картинку, которая была заранее подготовлена и хранилась в файле.

Перейдем к описанию двух популярных интегрированных систем: Симфони (Symphony, фирма Лотус Девелопмент) и Фреймуорк (Framework, фирма Эштон Тейт).

Интегрированная система Симфони. Многие плодотворные идеи в области программного обеспечения явились следствием анализа повседневных рутинных операций пользователя. Например, электронная таблица, завоевавшая такую популярность, была изобретена в результате осознания возможности реализации средствами компьютеров тех простых действий, которые каждый день выполняют тысячи служащих, составляя бесконечные таблицы, формы, заполняя бланки и анализируя тривиальные зависимости типа "суммарные расходы за неделю" или "средний уровень прибыли за пять лет".

Другая плодотворная идея, которая фактически привела к рождению интегрированных систем, столь же проста и очевидна: данные остаются данными, в какой бы форме они ни были представлены – в виде таблицы, или в виде графика, или просто в виде текста. Смена формы представления осуществляется на ЭВМ так же просто и естественно, как это делает пользователь вручную, выбирая наиболее удобную для анализа форму.

Именно эта идея однородности данных и инвариантности к форме представления последовательно выдержана при проектировании системы Симфони.

Данные в системе хранятся в общей области, которая называется рабочей областью. Данные в равной степени доступны всем пяти

прикладным пакетам: редактору текстов, электронной таблице, базе данных, графике и телекоммуникации. Доступ к данным осуществляется в многооконном режиме.

Симфони имеет развитую систему выдачи справочной информации. Если в некоторой ситуации пользователь не знает, что делать дальше, ему достаточно нажать клавишу "помощь". На экран при этом выдается информация о текущей ситуации (система "понимает", в какой точке находился пользователь в момент обращения за помощью) и перечень пунктов, по которым имеется более подробная информация.

Симфони имеет мощный текстовый процессор. Можно работать с блоками текста, осуществлять контекстный поиск и замену строк, выравнивать текст по правой границе и менять расположение границ на странице. Набрав текст в режиме текстового процессора, можно перейти в режим электронной таблицы и продолжить работу над текстом, вставляя в него числовую информацию с автоматическим пересчетом полей и т.д. Есть также широкий набор функций манипулирования строками: склейка, разбиение и т.д.

Основой базы данных в Симфони является понятие формы. База данных создается путем определения формы на экране. Форма состоит из описания полей, составляющих одну запись в базе данных. Описав форму, можно начать заполнение базы данных записями, заполняя указанные поля конкретной информацией. При этом форма играет роль шаблона, расположенного на экране, и система автоматически переходит к следующему полю по мере заполнения формы. Можно просматривать ранее созданную базу данных, которая хранится в виде файла на диске, и осуществлять поиск подмножества записей по указанному критерию. Простейший и очень удобный способ указания критерия заключается в определении шаблона поиска. На экран вызывается пустая форма и в ее поля заносятся ключевые значения. В результате поиска будут найдены все записи из базы данных, у которых указанные поля имеют эти значения. Можно использовать более сложные критерии поиска, используя широкий набор логических функций.

Симфони имеет многооконный интерфейс, что дает возможность работать одновременно с несколькими фрагментами рабочей области. Общение с каждой из них производится через окно. Окно создается специальным приказом и связывается с одним из пяти видов прикладных пакетов, имеющихся в системе.

Пакет телекоммуникации в Симфони осуществляет связь системы с внешним миром. Можно ввести в рабочую область любой файл в стандартном формате ASCII, созданный какой-либо программой. Команда "транслировать" осуществляет адаптацию файлов, созданных в форматах других известных систем, таких как Визикалк (Visicalc) или Дибейз–3 (dBase III), для их исполь-

зования в Симфони. Можно передавать файлы по телефонным линиям на другую персональную ЭВМ или на большую машину. Последняя возможность используется обычно для доступа к большим базам данных. Все действия по связи можно автоматизировать. Машина в указанное пользователем время сама наберет нужный номер, дождется ответа, примет нужную информацию и занесет ее в рабочую область на указанное место. Понятно, что все эти действия нужно запрограммировать. Для этого существует еще одна интересная возможность в системе – командный язык Симфони.

Командный язык является фактически макроязыком, на котором можно описать программу для исполнения в пакетном режиме. Кроме набора команд, доступного в режиме диалога, макроязык содержит развитую систему программных композиций, характерных для обычных языков программирования. К ним относятся операторы цикла и ветвления, условные выражения и т.д.

Симфони является развитием популярной интегрированной системы Лотус 1-2-3, которая разработана той же фирмой Лотус Девелопмент, признанным лидером в области создания интегрированных систем.

Как и всякая интегрированная система, Симфони требует много оперативной памяти – более 300 Кбайт. Память является платой за скорость. Вся рабочая область системы помещается целиком в оперативной памяти.

Система поставляется вместе со специальными обучающими дискетами, на которых записаны уроки по отдельным темам и демонстрационные задачи. Изучить систему и научиться с ней работать можно за пару дней, вообще не открывая руководств, а пользуясь только обучающими дискетами. Надо заметить, что такая возможность является косвенным показателем качества системы: простота обучения говорит о естественности набора приказов и разумности подбора средств, объединенных в систему.

Интегрированная система Фреймуорк. В основе системы, как и в случае Симфони, лежит определенная идея, которая заключается в следующем. Человек мыслит иерархически упорядоченными категориями. Например, при написании научной статьи он сначала продумывает ее общий план, а затем приступает к его реализации, развертывая отдельные положения. Если считать, что именно обработка идей, а не просто манипуляция с числами и текстами является основной деятельностью человека, сидящего за экраном персональной ЭВМ, то естественно попытаться создать систему, все средства которой ориентированы на эту деятельность.

Безусловно, речь не идет о создании системы искусственного интеллекта, которая будет подсказывать человеку идеи. Наоборот,

назначение системы состоит в том, чтобы снять с человека заботы о рутинных операциях, освободив его время и внимание для действительно творческой деятельности. В основе системы лежат средства эффективной иерархической организации понятий, порождаемых пользователем. При этом сами понятия, точнее их "материализация", покоится на привычном наборе средств, известных по другим интегрированным системам: произвольный текст, таблица данных, запись в базе данных, файл и т.д.

Принципиально новым моментом в системе Фреймуорк является то, что все эти средства рассматриваются с единой точки зрения: все они объявляются фреймами. Фрейм есть некоторый универсальный носитель упорядоченной информации, причем он, в свою очередь, может состоять из фреймов. Таким образом, разрешена иерархия фреймов, что соответствует иерархичности человеческого мышления. Перечисление списка фреймов, включаемых в некоторый объемлющий фрейм, эквивалентно разработке общего плана действий, за которым следует этап подробной проработки отдельных компонент. Например, при подготовке текста отчета в него можно заранее в подходящее место вставить фрейм графического представления какой-либо информации, не прерывая работы над основным текстом и не отвлекаясь на детали графики. В дальнейшем графический фрейм можно будет обдумать подробно и заняться его реализацией. Существенно то, что он уже занимает свое место в основном тексте и это высказывание верно на любом этапе его разработки, начиная с существования только его названия и до полной его разработки.

Заметим, что это в точности соответствует принятой среди программистов технологии развития программных систем" сверху вниз". Впрочем, Фреймуорк не навязывает такой технологии: можно начать с подробной проработки набора фреймов, а затем объединить их в фрейм-результат.

С технической точки зрения фреймы представлены в системе окнами, возникающими на рабочем поле, в которое превращается экран в момент входа в систему. Верхняя строка экрана занята меню. Меню содержит приказы общего характера, в частности приказ порождения фреймов. Как уже было сказано, существуют общепринятые типы фреймов: текст, таблица, база данных, графический фрейм, фрейм коммуникации и, наконец, новое понятие – фрейм-оглавление. Структура последнего вида аналогична оглавлению книги с нумерованными пунктами и подпунктами. Если пользователь выбирает из меню приказ "создать", на экране появляются альтернативы, и нужно указать тип создаваемого фрейма.

Интересно, что авторы системы весьма последовательны в своей привязанности к фреймам. Приказы в меню – это, оказывается,

тоже фреймы, поэтому выполнение приказа оформлено как вход в фрейм. В момент выполнения приказа "создать" на экране как бы вырастает содержимое фрейма, "подклеиваясь" снизу к его имени. Фрейм содержит альтернативы, и курсор останавливается на первой из них – порождении заголовка. Этим как бы подчеркивается, что разумно начать построение фрейма с определения в общих чертах его структуры. Если остановиться на предложенной альтернативе и нажать клавишу "исполнить команду", произойдет вход внутрь фрейма, соответствующего понятию заголовок, и на экране возникает пустой заголовок, готовый для заполнения информацией пользователя.

Справа на рабочем поле виден список логических имен внешних устройств, доступных системе. Если выбрать приказ меню "диск", то курсор попадает на первое из этих имен в столбце. Как и следует ожидать, имя это представляет "спящий" фрейм, который активизируется нажатием клавиши "исполнить команду", и на рабочем поле возникает внутренность фрейма, каковой является директорием (т.е. списком файлов) соответствующего диска. Если весь директорий не помещается в открытом окне, можно с помощью клавиш-стрелок двигаться вверх и вниз, открывая невидимые части директория. Если поместить курсор на имя файла и нажать "исполнить команду", на рабочем поле возникает новый фрейм, в котором виден текст файла. Можно начать его редактирование, так как возникший фрейм автоматически связан с текстовым процессором.

Фреймуорк имеет развитые средства ведения электронной таблицы, привычные пользователю по опыту работы с лучшими системами этого класса, такими как Лотус 1-2-3 и Симфони. Важной особенностью системы является то, что имя ячейки таблицы включает имя самой таблицы. Это означает, что в пределах одной таблицы можно ссылаться на информацию из другой таблицы.

Возможности программы ведения баз данных в системе соответствуют средним стандартам такого рода систем, выигрывая в каких-то одних деталях и проигрывая в других. Плохо, например, что допускается сортировка только по одному ключу. С другой стороны, тривиально делается изменение структуры записей. Достаточно сменить точку зрения на базу данных, рассматривая ее в виде электронной таблицы. Добавление новых полей или удаление существующих полей после этого не представляет труда. Очень важно то, что система Фреймуорк полностью совместима с другой продукцией той же фирмы Эштон-Тейт – одной из лучших существующих систем ведения баз данных для персональных компьютеров – Дибейз-3.

Средства телекоммуникации являются слабым местом системы. Язык, который использует пакет телекоммуникации, отличается от

языка остальных компонент системы, что создает большие неудобства для пользователя. Система использует протокол XMODEM и позволяет обмениваться файлами в двоичном и ASCII форматах.

Система Фреймуорк имеет встроенный язык высокого уровня Фред, на котором можно программировать различные надстройки над системой, ориентированные на пользователя. Имеется также возможность вводить макроопределения, т.е. последовательности команд системы, которые потом вызываются нажатием одной клавиши. Вместе с системой поставляется небольшая библиотека полезных макросов.

Еще одним мощным средством расширения является фрейм доступа к операционной системе. Вход в фрейм выводит на уровень операционной системы. При этом можно выполнять любые ее команды. Например, можно запустить какую-то свою программу и результаты ее работы передать в систему Фреймуорк. Чтобы вернуться в систему, достаточно дать команду "выход". Эта возможность делает систему очень близкой по свойствам к операционным оболочкам, основной целью которых является интегрирование автономно разработанных программ.

Система поставляется с обучающими дискетами, о которых следует сказать особо. Легкость обучения является одним из важнейших критериев при выборе системы. Программа обучения системе Фреймуорк является одним из лучших образцов в своем классе. Она представляет собой не сухое руководство, а увлекательную игру, от которой трудно оторваться, не дойдя до конца.

Задача 4

Распределение посевных площадей

Имеется сельскохозяйственный район с известным перечнем посевных площадей, на которых можно выращивать злаки из заданного множества. Требуется сделать систему, которая позволит оценивать экономическую эффективность различных вариантов распределения злаков по посевным площадям. Существенным требованием к системе является наличие в ней средств графического представления результатов выбора вариантов, например в виде карты района с указанием распределения злаков по посевным площадям.

Решение первых трех задач пожно было провести средствами интегрированных систем. Данная задача относится к другой категории. Ее отличительной чертой является то, что в основе решения лежат достаточно сложные математические модели. Эти модели позволяют произвести расчет урожайности злаков с учетом многочисленных факторов, таких как влияние полива и внесения удобрений, состояние почвы, готовность и количество сельскохозяйст-

венных машин, возможность их быстрого перемещения с участка на участок (а это в свою очередь зависит от наличия и состояния дорог) и т.д.

Трудно найти такую прикладную программу, которая точно соответствовала бы поставленной задаче. Собственно говоря, нет одной задачи, а есть совокупность взаимосвязанных задач, постановка которых возникает и может меняться по ходу решения. Поэтому трудно в начале этого процесса угадать все средства, которые могут понадобиться при решении. Какая-то часть анализа может быть проведена, например, средствами электронной таблицы. Затем, возможно, потребуется применить программы моделирования динамических процессов, которые описываются дифференциальными уравнениями. Вряд ли удастся обойтись также без программ решения оптимизационных задач различного класса. При описании задачи было сказано, что важнейшим элементом решения является разработка удобных графических форм выдачи результатов на экран компьютера. Здесь явно недостаточно будет простых средств деловой графики, которые есть в интегрированных системах. На экран требуется выдавать довольно сложные картинки типа географической карты с указанием рекомендуемых участков для посева. Значит, надо включать в систему прикладные программы так называемой демонстрационной графики.

Таким образом, задача носит комплексный характер. Вообще говоря, таким свойством обладает практически любая реальная задача. При этом каждый отдельный аспект анализа задачи достаточно прост, по крайней мере, на начальной стадии решения. Проблема заключается не в том, что нет средств анализа этих аспектов – их обычно вполне достаточно. Трудность заключается в общей организации работ, в стыковке отдельных этапов, т.е. в "наращивании" технологии решения. Для таких задач требуются очень динамичные средства решения, которые легко настроить под задачу или заменить на другие. Средства решения, практически, должны создаваться одновременно с самим процессом решения.

Персональный компьютер обладает всеми необходимыми для этого свойствами: он прост в общении, для него разработано огромное количество прикладных программ, его можно настраивать "под себя", отбирая в процессе работы те средства, которые больше нравятся. Удобным средством для такой настройки являются операционные оболочки.

Операционные оболочки. В отличие от интегрированных систем, в которых интеграция определена априори, операционные оболочки являются средством осуществления интеграции в объеме, определяемом самим пользователем. Появление на программном рынке этих систем явилось отражением того факта, что интегрированные системы слишком статичны и не могут удов-

летворить всех пользователей. Многим из них просто не хочется платить лишние деньги за компоненты системы, которые они не собираются использовать. Вместе с тем просто купить нужный набор программ недостаточно, если нет средств их стыковки. Можно, конечно, купить нужный набор программ из единого семейства, но остается проблема их совместного использования с программами собственной разработки. Эти проблемы решаются средствами операционных оболочек.

Причину возникновения этих систем и их бурного развития следует искать в самом феномене персональных ЭВМ.

Появление персональных ЭВМ привело к вовлечению в сферу программистской деятельности огромного количества людей, а значит – и новых идей. Прогресс в это области настолько велик, количество новых, все более "умных" программ во всех прикладных областях так быстро растет, что средства общесистемного программного обеспечения, включая операционные системы, просто не успевают держаться на уровне новых требований. Системные средства, будучи по своей природе более консервативными, чем прикладные программы, начинают тормозить их развитие. Например, удобные средства многооконного интерфейса, привычные пользователю по многим прикладным системам, только начинают проникать на уровень операционных систем. Проблему решает новая категория программных средств – операционные оболочки, которые являются расширениями операционных систем и по своей динамичности занимают промежуточное положение между операционными системами и прикладными пакетами.

По мере отработки и стабилизации идей они "опускаются" на уровень операционной системы, освобождая в операционных оболочках место для реализации новых возможностей, подсказанных развитием прикладных программ. Этот процесс аналогичен взаимодействию слоев аппаратного и программного обеспечения. Аппаратура берет на себя все больше функций, которые традиционно выполнялись программным слоем.

Операционные оболочки обеспечивают две основные функции интеграции прикладных пакетов:

включение новых прикладных пакетов в систему и организация взаимодействия между ними;

организация многооконного интерфейса с пользователем в процессе управления задачами.

Опишем основные принципы реализации этих функций в нескольких системах, таких как Топ Вью (Top View, фирма IBM), Джем (GEM, фирма Digital Research), Уиндоус (Windows, фирма Microsoft), Конкаррент Дос (Concurrent PC DOS, фирма Digital Research), Прогрэммерс Ассистант (Programmer's Assistant, фирма Xerox). Начнем описание с принципов организации многооконного интерфейса.

Многооконный интерфейс

Первые работы по организации многооконного интерфейса с пользователем были выполнены в исследовательской лаборатории Парк (PARC – Palo Alto Research Center) фирмы Ксерокс в Калифорнии в начале семидесятых годов. Смысл многооконного интерфейса заключается в разработке новых средств общения с машиной, ориентированных в первую очередь на непрофессионального пользователя.

Особенно широкое распространение средства многооконного интерфейса получили в связи с появлением персональных ЭВМ. Выше уже обсуждалось применение многооконного интерфейса в текстовых редакторах и электронных таблицах. Важную роль в популяризации многооконного интерфейса сыграли машины Макинтош и Лиза фирмы Эппл (Apple), в которых он был заложен в основу организации всего программного обеспечения этих машин. Идея многооконного интерфейса оказалась настолько удачной, что сегодня практически все новые разработки программного обеспечения ориентированы на этот интерфейс.

Ниже излагается организация многооконного интерфейса в различных системах по перечню основных параметров, определяющих его свойства.

Дисциплина появления и переключения окон. Существует два различных подхода. Первый подход основан на полной свободе со стороны пользователя в порождении окон и их расположении на экране. Этот подход реализован, например, в Симфони и Фреймуорк. Окна порождаются на рабочем поле в любом количестве и в любом месте. В обеих системах в процессе работы экран дисплея часто приобретает такой вид, который имеет поверхность письменного стола с хаотически набросанными на нем листами бумаги.

Разработчики системы Уиндоус придерживаются другого подхода. Основная его идея заключается в разумной регламентации работы пользователя с окнами. Не допускается перекрытие окон на экране. Можно, например, разбить экран на четыре одновременно существующих окна. При увеличении размера одного из окон оно может наложиться на другое окно. Это окно при этом снимается с экрана и превращается в небольшую пиктограмму, "спящую" в правом нижнем углу экрана. В дальнейшем можно активизировать это окно, и тогда оно вытеснит с экрана предыдущее окно.

Переключение с одного окна на другое трактуется по-разному в зависимости от наличия или отсутствия в системе многозадачного режима. Заметим, что реализация многозадачного режима, при котором пользователь может решать несколько прикладных задач одновременно, является одной из важных функций операционных

оболочек. В случае однозадачного режима переключение на другое окно означает прерывание текущей задачи, связанной с активным в данный момент окном, и активизация другой задачи. При наличии многозадачного режима переключение окон не означает обязательного прерывания задачи, связанной с текущим окном. Например, в операционной системе Конкаррент Дос, которая допускает одновременное решение до четырех задач, все они могут выполняться параллельно и каждая связана с одним окном. В Топ Вью и Уиндоус вообще нет ограничений на количество одновременно решаемых задач.

С т р у к т у р а о к н а. В большинстве систем позиция окна на экране задается пользователем или по крайней мере может меняться по его усмотрению.

Можно изменять другие характеристики окна – его размеры, цвет фона и цвет текста в окне и т.д. Существует несколько способов изменения характеристик. Самый простой (и не самый удобный) способ состоит в задании численных значений параметров окна при выполнении соответствующего приказа. Более удобный способ реализован, например, в Конкаррент Дос. Для изменения положения или размера окна достаточно выбрать соответствующий приказ меню и после этого нажать, например, клавишу "стрелка вправо". Окно будет смещаться вправо до тех пор, пока нажата клавиша. Если был выбран приказ изменения размера окна, оно начнет увеличиваться за счет движения его правой границы вправо. В системе Уиндоус реализована другая возможность, ориентированная на применение устойства "мышь". В углах окна есть "угловые скобки". Позиционирование этих скобок в новое положение на экране с помощью устройства "мышь" заставляет окно менять свои размеры.

В системах с многооконным интерфейсом активно используются так называемые пиктограммы, т.е. небольшие картинки, вид которых поясняет смысл команды меню. Например, команда уничтожения файла изображается в виде корзины для бумаг. Пиктограммы в основном используются в системах, снабженных устройством "мышь". Часто сам курсор изображается на экране в виде пиктограммы, которая меняется в зависимости от ситуации и дает о ней дополнительную информацию. Популярность машин Макинтош и Лиза фирмы Эппл во многом объясняется тем, что они полностью ориентированы на многооконный интерфейс с активным использованием "мыши" и пиктограмм. Идея создания персональной ЭВМ, которой могли бы эффективно пользоваться люди, не имеющие никакого навыка общения с ЭВМ, была положена в основу разработки этих машин.

Интерфейс, реализованный на машинах Макинтош, стал настолько популярен, что разработчики программного обеспечения стали

создавать аналогичные программы для других персональных компьютеров. Одна из наиболее интересных работ в этой области сделана фирмой Диджитал Ресерч. Фирма разработала систему Джем, которая реализована на многих популярных моделях персональных компьютеров.

Система Джем состоит из нескольких программ. Основная из этих программ называется Джем Коллекшн (GEM Collection) и включает три части:

GEM Desktop – программа организации Макинтош-подобного интерфейса, основанного на окнах, меню, пиктограммах и использовании устройства "мышь";

GEM Write – текстовый редактор;

GEM Paint – программа-художник, с помощью которой можно создавать произвольные картинки в цвете.

Важной особенностью системы является то, что программы GEM Write и GEM Paint можно использовать вместе, совмещая произвольным образом текст и картинки.

GEM Desktop обеспечивает работу с экраном в многооконном режиме с активным использованием пиктограмм. На экране располагаются пиктограммы, соответствующие имеющимся внешним устройствам. Если подвести курсор к пиктограмме флоппи-диска, на экране появляется окно, в котором в виде пиктограмм изображены все файлы, имеющиеся на этом диске. Чтобы уничтожить файл, достаточно просто указать на него курсором и затем переместить его пиктограмму в мусорную корзину, которая изображена в углу экрана. Текстовые файлы изображаются специальными пиктограммами, и можно сделать так, чтобы выборка любого текстового файла с помощью курсора автоматически приводила к вызову текстового редактора для его редактирования. Указание на пиктограмму поддиректория приводит к входу в этот поддиректорий. Указание на пиктограмму исполняемой программы приводит к ее запуску на счет.

Система Джем находится в стадии быстрого развития. В нее постоянно включаются новые интересные программы, такие как GEM Draw для чертежной графики в цвете, GEM Graph для деловой графики и т.д. Система Джем в отличие от Топ Вью позволяет работать с прикладными программами, ориентированными на графическую выдачу информации. Топ Вью предназначена для работы с прикладными программами, которые используют только алфавитно-цифровой режим выдачи информации на экран. Учитывая, что графика является перспективной формой организации интерфейса это можно считать серьезным недостатком системы.

В Топ Вью включена специальная программа "конструктор окон". Она позволяет создать окно, указав его размеры, цвет, положение на экране. Можно провести структуризацию окна, оп-

ределив в нем различные поля с их характеристиками для ввода и вывода числовой и символьной информации.

Дисциплина взаимодействия окно–процесс. Окно есть средство управления некоторым вычислительным процессом. Из многих окон, существующих в данный момент в системе, одно окно является активным в том смысле, что с его помощью можно управлять в диалоге связанным с ним процессом. Остальные окна пассивны – с ними нет двусторонней диалоговой связи. Это отнюдь не означает, что связанные с ними процессы пассивны. В системах с многозадачным режимом эти процессы могут идти параллельно. Именно так работают системы Конкаррент Дос и Топ Вью, в которых можно так расположить окна на экране, чтобы была видна выдача из всех работающих процессов одновременно. Система Джем не допускает многозадачного режима.

В каждой многооконной системе общего назначения дисциплина взаимодействия окно–процесс согласована с действующими в ней принципами порождения окон и переключения между окнами. Например, в системе Уиндоус не допускается перекрытие окон на экране. Если активное окно накладывается сверху на другое окно, то последнее исчезает с экрана.

В операционных оболочках заранее не известно, какие программы будут запущены в окнах. Может оказаться, что какая-то программа выдает информацию на экран дисплея путем прямого доступа к памяти, минуя средства операционной системы. Это, естественно, может полностью нарушить всю картину на экране. В системе Concurrent средство борьбы с такой ситуацией заключается в следующем. Если заранее известно, что программа обладает таким свойством, то при связывании ее с окном можно выполнить специальную команду "задержать выполнение". До тех пор, пока программа связана с активным окном, она сама распоряжается формой выдачи информации на экран. При переключении на другое окно программа будет автоматически переведена в пассивное состояние.

В системе Уиндоус все программы должны использовать специальные программные средства выдачи информации на экран. Уиндоус имеет встроенный программно-реализованный графический интерфейс, что делает ее в высокой степени независимой от конкретной аппаратуры. Интерфейс графического устройства GDI (Graphic Device Interface) является пакетом графических подпрограмм, который играет роль посредника между прикладной программой и аппаратурой. GDI – это язык абстрактного графического устройства. Прикладная программа выдает приказы на этом языке, а драйвер конкретного графического устройства переводит их в систему команд, понятных аппаратуре.

В Топ Вью при включении программы в систему требуется

указать тип ее взаимодействия с экраном, и система учтет эту информацию при организации многозадачного режима работы.

Включение новых пакетов и взаимодействие между ними. Во всех операционных оболочках реализованы гибкие средства включения новых прикладных программ в систему. В качестве примера опишем, как это делается в Топ Вью. На экран можно вызвать список прикладных программ, включенных в систему. Ниже этого списка располагается меню, в которое входит команда "включить программу". При выполнении этой команды система просит ввести следующую информацию о программе:

имя программы, которое будет включено в список имеющихся прикладных программ;

имя файла, в котором хранится программа;

требуемый объем памяти для работы программы;

использует ли программа прямой доступ к памяти экрана;

использует ли программа прямой доступ к буферу клавиатуры;

использует ли программа арифметический процессор 8087.

Вся эта информация о программе запоминается в специальном файле и используется системой при запуске программы на счет. Работающую программу можно в любой момент прервать и запустить другую программу, а затем продолжить счет первой программы с прерванного места.

Одновременное управление несколькими задачами через систему окон на экране дисплея представляет особый интерес при наличии средств их информационного взаимодействия. Эти средства реализованы во всех операционных оболочках. В Топ Вью, например, параллельно работающие прикладные программы могут обмениваться информацией в виде сообщений или путем организации потоков данных. Уиндоус обеспечивает обмен информацией между окнами на одном из трех уровней: пересылка файлов в формате ASCII, двоичных образов файлов и файлов в специальном формате, разработанном фирмой Микрософт для прикладных программ. Протокол передачи данных система выбирает сама. Каждая программа сообщает системе, какой метод передачи данных для нее приемлем. Система по этой информации определяет протокол максимального уровня, приемлемый для партнеров по обмену, и применяет его. Еще одна интересная возможность по связи окон состоит в следующем. Уиндоус позволяет одному окну сформулировать "заказ на информацию" из другого окна. Это похоже на задание значения в ячейке электронной таблицы в виде формулы от значений других ячеек.

Специальные команды позволяют выделить кусок текста в одном окне и перенести его в другое окно на указанную позицию. Если проанализировать эти действия, то нетрудно понять, что для

их реализации требуются средства взаимодействия между окнами не только по информации, но и по управлению. Естественной представляется, например, следующая организация окон. Одно из окон играет роль меню, содержащего общеупотребительные команды, исполнение которых имеет смысл в любом окне. Например, команды редактирования текста или выделения фрагментов текста. Выбрав команду редактирования, можно переместить курсор в любое окно на экране и отредактировать его текст.

Приведем пример использования этой формы взаимодействия окон в том виде, как она реализована в системе Прогрэммерс Ассистант.

Система Прогрэммерс Ассистант является многооконной системой, ориентированной на программистов, использующих язык Лисп. Система имеет несколько специализированных типов окон, обеспечивающих все технологические этапы работы программиста. В одном окне можно набирать тексты новых функций, в другом – делать структурную распечатку этих текстов, облегчающую их отладку, в третьем – анализировать процесс вычисления функций по всем его этапам с возвратом при необходимости на любой этап и повторения вычислений с указанной точки и т.д.

Характерной особенностью системы является возможность так называемого контекстного переключения. Программист может прервать процесс набора текста новой функции в произвольном месте и с помощью окна-меню переключиться на другое окно, связанное с библиотекой имеющихся в системе лисп-программ. Просмотрев интересующую его программу, он возвращается в первое окно и продолжает работу над функцией с прерванной точки. Он может также выделить фрагмент текста из библиотечной функции и переместить этот фрагмент в позицию курсора в первом окне. Поскольку система включает средства телекоммуникации, можно запросить текст функции или ее фрагмента у коллеги, работающего в соседней лаборатории. Получив текст, который будет напечатан в окне сообщений вместе с некоторой информацией о факте связи, фамилией отправителя и т.д., можно выделить нужный блок текста и переместить его в текущую позицию курсора рабочего окна.

Этот пример демонстрирует еще один аспект операционных оболочек – наличие в них средств связи с другими машинами по линиям телекоммуникации. Это естественное развитие основной идеи – одновременного управления многими процессами и "перекрестного" использования информации между ними.

В системе Прогрэммерс Ассистант есть два типа окон для связи: окно посылки сообщения и окно приема сообщения. Если программист получил сообщение, информация об этом возникает в его рабочем окне. Можно прервать работу и посмотреть сообще-

ние или продолжать работу и обратиться к нему позже. Вообще, для многооконных систем характерен асинхронный характер действий с переключениями между работами, возвратами и т.д. Это, естественно, требует развитой системы обработки прерываний на уровне операционной системы.

Следует отметить, что пока еще недостаточно разработаны принципы структурной организации систем такого рода. Ситуация напоминает ту, которая возникла в программировании при неограниченном использовании оператора перехода goto. Глядя на экран с хаотическим нагромождением окон, часто трудно бывает восстановить предысторию их создания и взаимодействий. Ситуация тем более усложняется, что в многооконных системах диалоговые контексты динамичны в отличие от статических текстов программ, которые по этой причине легче анализировать. Очевидно, необходима некоторая регламентация деятельности пользователя со стороны системы. Полная свобода переключения диалоговых контекстов создает только видимость гибкости, на самом деле снижая производительность.

В заключение ответим на ряд вопросов, которые, возможно, возникли у читателя.

Кто может работать на персональном компьютере?

Ответ на этот вопрос дать проще всего: кто угодно, начиная с детей младшего школьного возраста и кончая профессиональными программистами, разработчиками программного обеспечения. Интересно, что дети гораздо быстрее обучаются работе на персональном компьютере, чем взрослые, воспринимая этот процесс скорее как увлекательную игру, чем как серьезное занятие.

Что можно делать на персональном компьютере?

На персональном компьютере можно обрабатывать информацию: создавать и редактировать тексты, принимать или передавать информацию на другой компьютер, производить вычисления, рисовать картинки, разучивать гаммы на клавиатуре фортепиано, изображение которой появляется на экране и реагирует звучащей нотой на перемещение курсора по клавиатуре. Можно в паре с компьютером заниматься творческой деятельностью: проектировать автомобиль, анализировать план-график выполнения работ на заводе, играть в шахматы или нарды. Можно обучаться какой-либо профессии, например изучать сам персональный компьютер под его же руководством. Можно изучать школьный курс химии, физики, математики и даже истории. При этом школьник не рискует поджечь дом, проводя химический опыт, и может видеть процесс развития реакции в виде цветного мультфильма на экране компьютера.

Можно решать задачи из различных прикладных областей. Для "непрограммирующего профессионала" (т.е. специалиста в какой-

то прикладной области, который сам не умеет программировать) или для начинающего пользователя, изучающего приемы использования персонального компьютера на примере конкретных задач, возможность решения задач определяется тем, есть ли на рынке программного обеспечения прикладные программы, предназначенные для решения задач требуемого класса. Как уже говорилось выше, число таких прикладных программ очень велико и растет с каждым днем.

Персональные компьютеры дают возможность по-новому подойти к решению проблемы автоматизации управленческой деятельности. Руководитель, имеющий на своем рабочем столе персональный компьютер, получает возможность наращивать свою собственную, личную технологию обработки информации, которая включает в себя собственно информацию в виде базы данных, набор прикладных программ для ее обработки и набор неформальных процедур, приемов, правил, отражающих его предпочтения и привычки. Очевидны преимущества такого персонального подхода, в первую очередь с психологической точки зрения. Человек полностью управляет всем процессом: подбирает для себя информацию, ищет наиболее эффективные с его точки зрения приемы ее обработки, пополняет библиотеку прикладных программ. Это повышает уверенность руководителя в качестве информации, на основании которой он принимает решения. Появляется возможность более целенаправленно выдавать задания на подготовку информации из внешних источников: централизованных банков данных, автоматизированных систем управления, от руководителей подчиненных подразделений и т.д.

Трудно ли научиться работать на персональном компьютере?

Это естественный и самый важный вопрос для начинающего пользователя. И все же вместо ответа хочется задать аналогичный "простой" вопрос: трудно ли научиться играть на скрипке? Смотря что и смотря как играть. Одно можно сказать с уверенностью: важнейшим критерием для разработчиков персональных компьютеров и их программного обеспечения является простота и удобство работы для пользователя-непрофессионала. Настоящий опыт эффективного использования компьютера, как и в любом деле, приходит со временем и требует определенных усилий. Это естественно, если учесть, что персональный компьютер с его программным обеспечением является неисчерпаемым источником новых интересных возможностей. Овладение ими представляет собой творческий процесс: человек реализует свои идеи средствами компьютера.

Что касается исходного обучения, то можно сослаться на критерий, ставший уже традиционным для персональных компьютеров. Считается, что научиться работать на компьютере не сложнее,

чем научиться управлять легковым автомобилем, когда рядом с учеником сидит инструктор. Роль инструктора берет на себя сам компьютер. Выше уже говорилось о том, что практически все прикладные программы снабжаются, кроме печатных инструкций и учебников, специальными обучающими дискетами. Работа с таким "живым учебником" позволяет изучить возможности прикладной программы, вообще не прибегая к помощи инструкций и книг.

В процессе работы с прикладной программой можно пользоваться "электронной скорой помощью", которая вызывается нажатием специальной клавиши и немедленно прибывает, готовая оказать услугу, т.е. напомнить какие-либо правила, сообщить список команд или смысл функциональных клавиш, объяснить ошибку и т.д.

Как известно, на вопрос "умеете ли вы играть на скрипке?" дилетант отвечает "не знаю, не пробовал". Лучший способ понять, что такое персональный компьютер – это попробовать на нем поработать.

Операционная оболочка	–	программа, занимающая промежуточное положение между операционной системой и прикладными пакетами; ее цель – интеграция прикладных пакетов.
Операционная система (ОС)	–	важнейшая часть программного оснащения компьютера, обеспечивающая управление всеми аппаратными компонентами и позволяющая отделить остальные классы программ от непосредственного взаимодействия с аппаратурой.
Оперативная память, или оперативное запоминающее устройство (ОЗУ)	–	устройство, где размещаются во время исполнения программы, а также используемые ими данные; оперативная память характеризуется более высокой скоростью записи и чтения и меньшим объемом, чем внешняя память; при выключении машины содержимое оперативной памяти не сохраняется.
Постоянное запоминающее устройство (ПЗУ)	–	разновидность памяти ПЭВМ, содержимое которой постоянно (сохраняется при выключении машины); запись информации в ПЗУ невозможна, а чтение может производиться с высокой скоростью; в ПЗУ обычно находятся программы и данные, обслуживающие работу аппаратуры.
Прикладной пакет	–	комплекс взаимосвязанных программ для решения задач определенного класса.
Принтер	–	устройство для печати на бумаге информации, передаваемой из памяти персонального компьютера.
Программа	–	последовательность действий (команд, операторов), записанная на специальном языке и предназначенная для выполнения компьютером.
Системный блок	–	корпус, в котором находятся основные электронные компоненты персонального компьютера, а также блок питания; в нем могут размещаться внешние накопители, а иногда в него встраиваются также дисплей и клавиатура.
Текстовый редактор		программа для подготовки и обработки текстовой информации, которая позволяет вводить символы (буквы, цифры и др. знаки) с клавиатуры и осуществлять различные действия по изменению (редактированию) текстов под управлением пользователя.
Телекоммуникации средства	–	программные и аппаратные средства, позволяющие передавать информацию от одного компьютера к другому.
Транслятор		особая программа, назначение которой состоит в преобразовании текста, записанного человеком-программистом на языке программирования, в машинный код, который может быть исполнен компьютером.
Файл	–	именованная область внешней памяти для хранения программ и данных для их работы; в файлах могут содержаться произвольные текстовые документы и числовые данные, закодированная табличная, графическая и любая другая информация.
Функциональные клавиши	–	группа клавиш на клавиатуре ПЭВМ, не имеющих постоянного, закрепленного назначения; ФК могут "программироваться", т.е. приобретать по воле программиста определенный смысл в прикладных и системных программах.
Электронная таблица	–	программа обработки числовой и текстовой информации, упорядоченной в виде таблицы с именованными строками и столбцами.
Язык программирования	–	формальная система для записи алгоритмов в виде программ.

КРАТКИЙ СЛОВАРЬ ТЕРМИНОВ

База данных – информация, упорядоченная в виде набора элементов (записей) одинаковой структуры; для обработки записей используются специальные программы, позволяющие упорядочивать записи, делать выборки по указанному правилу; обычно базой данных называют и информацию, и программы ее обработки.

Байт – общепринятая единица измерения информации, используемая для указания объема памяти, скорости передачи информации и других характеристик ЭВМ. Один байт состоит из 8 битов. При представлении символьной (текстовой) информации каждая буква, цифра или знак занимает 1 байт; 1 Кбайт (произносится "килобайт") равен 1024 байтам, но для простоты часто говорят, что 1 Кбайт – это "тысяча байт"; 1 Мбайт (произносится "мегабайт") равен 1024 Кбайтам, но часто говорят, что 1 Мбайт – это "миллион байт".

Бит – двоичный разряд, элементарная единица информации, принимающая значения 0 или 1.

Внешние накопители – устройство для постоянного хранения информации (программ и данных): магнитные ленты, гибкие и жесткие магнитные диски.

Деловая графика – программа подготовки графической информации и выдачи ее на экран или принтер; обычно используется ограниченный набор вариантов: гистограммы, линейные графики, круговые диаграммы.

Дисплей – устройство визуального отображения информации; на экран дисплея выводятся тексты и графические изображения.

Знакоместо – элемент экрана дисплея, в котором может быть изображен ровно один знак: буква или цифра; типичный экран ПЭВМ содержит 25 строк по 40 или 80 знакомест в каждой.

Интегрированная система – программа, включающая несколько взаимосвязанных прикладных пакетов; обычно содержит текстовый редактор, электронную таблицу, базу данных, деловую графику, средства телекоммуникации.

Каталог файлов – логический раздел на внешнем накопителе, объединяющий группу файлов и хранящий информацию об имени, объеме, дате и времени создания (или последнего изменения) файла.

Клавиатура – устройство ввода текстов, чисел и управляющей информации в память персонального компьютера; внешне похожа на клавиатуру обычной пишущей машинки, но имеет дополнительные группы клавиш для расширения возможностей управления компьютером.

Команды ОС – текстовые приказы, вводимые пользователем с клавиатуры и содержащие обращения к различным функциям ОС.

Курсор – мигающий или выделенный другим способом значок на экране дисплея, который обычно указывает позицию, где отображается очередной вводимый с клавиатуры символ.

Микропроцессор – электронная микросхема, обеспечивающая выполнение арифметических, логических и управляющих операций, заданных программой в машинном коде; тип микропроцессора определяет внутреннюю архитектуру ПЭВМ.

Многооконный интерфейс – вид диалогового взаимодействия пользователя с программами, при котором каждой программе отводится прямоугольная область на экране, называемая окном.

СОДЕРЖАНИЕ

	Предисловие	3
Г.Б. Кочетков	Персональные компьютеры на работе и дома	7
В.М. Брябрин	Как устроен и работает персональный компьютер	15
Г.В. Сенин	Приглашение к программированию	60
В.П. Мазурик	Прикладные системы и решение задач	102

ПЕРСОНАЛЬНЫЕ
КОМПЬЮТЕРЫ:
ИНФОРМАТИКА
ДЛЯ ВСЕХ

Сборник статей

Утверждено к печати
редколлегией серии
научно-популярных
изданий
Академии наук СССР

Редактор
А.А. Боровая

Художник
А.М. Драговой

Художественный редактор
Н.А. Фильчагина

Технический редактор
Н.А. Торгашова

Корректор
Е.Н. Сафронникова

Набор выполнен в издательстве
на наборно-печатающих автоматах

ИБ № 35554

Подписано к печати 25.11.86. Т – 22822
Формат 84 X 108 1/32
Бумага офсетная № 2
Гарнитура Пресс-Роман. Печать офсетная
Усл.печ.л. 7,6. Усл.кр.-отт. 11,9
Уч.-изд.л. 8,9. Тираж 243000 экз.
Тип. зак. 92. Цена 55 коп.

Ордена Трудового Красного Знамени
издательство "Наука" 117864 ГСП-7,
Москва В-485, Профсоюзная ул., д. 90

Ярославский полиграфкомбинат Союз-
полиграфпрома при Государственном ко-
митете СССР по делам издательств,
полиграфии и книжной торговли
150014, Ярославль, ул. Свободы, 97

55 коп.

КНИГИ СЕРИИ

"КИБЕРНЕТИКА – НЕОГРАНИЧЕННЫЕ ВОЗМОЖНОСТИ И ВОЗМОЖНЫЕ ОГРАНИЧЕНИЯ"

ВЫШЛИ ИЗ ПЕЧАТИ

Возможное и невозможное в кибернетике (1963)
Кибернетика ожидаемая и кибернетика неожиданная (1968)
Кибернетика. Итоги развития (1979)
Кибернетика. Современное состояние (1980)
Кибернетика. Перспективы развития (1981)
Кибернетика. Дела практические (1984)
Кибернетика живого. Биология и информация (1984)
Кибернетика живого. Человек в разных аспектах (1985)
Кибернетика и ноосфера (1986)
Кибернетика, ноосфера и проблемы мира (1986)
Кибернетика. Микрокалькуляторы в играх и задачах (1986)
Кибернетика. Становление информатики (1986)

ГОТОВИТСЯ К ПЕЧАТИ

Информатика и прогресс науки



«НАУКА»